Para Programadores MSX tu fiel compañero.

Este no es un libro para el principiante novato, al contrario, introduce al programador MSX "dentro" del ordenador, para mostrarle exactamente como funciona y como obtener el máximo partido.

CHRIS BURKINSHAW es un programador con inmensa experiencia, que acaba de completar un libro titulado "Más allá de BASIC en tu Commodore 64". Asímismo, ROSS GOODLEY es muy respetado en el mundo de la Informática, siendo especialmente conocido por el éxito de sus programas de juegos escritos para Alligata Software Ltd.

Los autores han dividido su libro en dos partes.

La primera parte cubre el diseño del sistema, como el vocabulario de BASIC se relaciona con los ordenadores MSX, y una introducción al código máquina. Encontrará explicaciones detalladas sobre la organización de la memoria, los modos de pantalla y los circuitos de video y sonido.

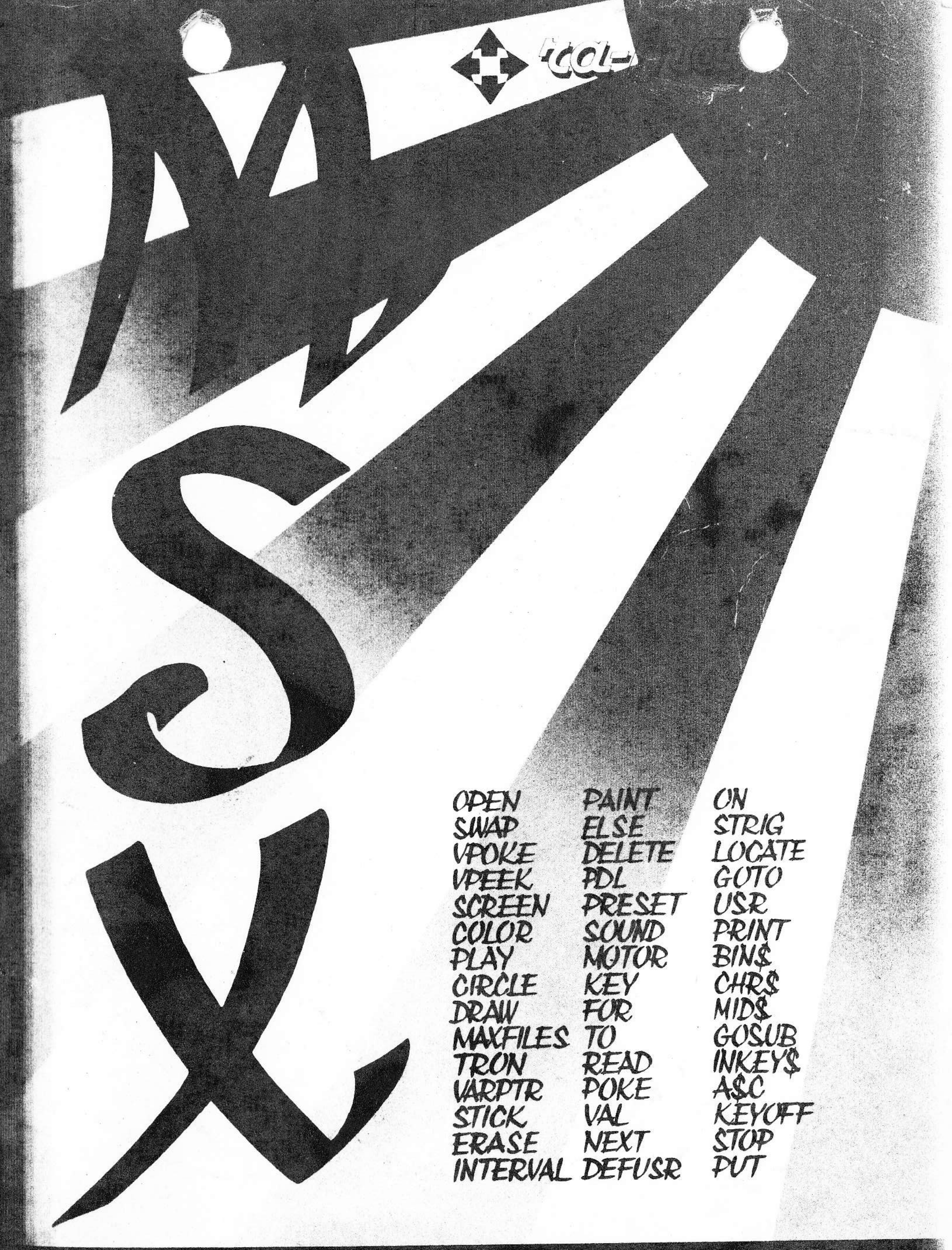
La segunda parte se dedica al uso del lenguaje ensamblador en el sitema MSX. Los principales apartados incluyen los puntos:

- El procesador de pantalla de video.
- El circuito de sonido AY-3-8910.
- Entradas/Salidas.

Tanto si necesitas escribir programas de alta eficacia o solamente quieres conocer como trabaja tu ordenador, este libro será inestimable. Pensamos que rápidamente se convertirá en el texto base para aquellos programadores que quieren hacer algo más que escribir programas en BASIC.

ca-ma
CHIQUINQUIRA, 28 (COCUY)
28033 MADRID

GUIA DEL PROGRAMADOR MSX • ca-ma



GUIA DEL PROGRAMADOR MSX

Guia del Programador

MSX

C. I. BURKINSHAW Y R. GOODLEY

ta-ma
CHIQUINQUIRA, 28 (COCUY) - 28033 MADRID.
TELEF.: 764 50 95

A nuestros padres,

sin los que no hubiera sido posible este libro

CONTENIDO

1. INTRODUCCION 9
   - Generalidades 9
   - Organización de la Memoria 10
   - Puertos de Entradas/Salidas 12
   - Conexión a Cassette 12
   - Modos de Pantalla 13
   - El VDP (Procesador de Pantalla) 13
   - Estructura de Pantalla en el VDP 15
   - Gráficos de Alta Resolución 18
   - Color 18
   - Sprites (Figuras Moviles) 20
   - Circuito de Sonido 22

2. BASIC MSX 23
   - Generalidades 23
   - Variables 24
   - Funciones 25
   - Instrucciones Gráficas 26
   - Instrucciones Generales 26
   - Modos de Texto 26
   - Manipulación de la RAM de Video 29
   - Ejemplo: Texto en Varios Colores 30
   - Programa Ejemplo: Juego de Caracteres 31
   - Sprites: Figuras Móviles 32
   - Programa Ejemplo: Diseño de Sprites 37
   - Gráficos de Alta Resolución 38
   - Programa Ejemplo: Tablero de Apuntes 41
   - Sonido 42
   - Almacenamiento de Programa 44

3. VOCABULARIO BASIC MSX 47

4. LENGUAJE MAQUINA Z-80 74
   - Microprocesadores 74
   - Organización del Sistema 75
   - Notación Binaria y Hexadecimal 76
   - Operaciones Lógicas 79

La Arquitectura del Z-80 80
El conjunto de instrucciones del Z-80 82
Operaciones de carga de 8 y 16 bits 84
Instrucciones Aritméticas de 8 y 16 bits 85
Operaciones Lógicas de 8 bits 86
Operaciones de rotación y desplazamiento 87
Operaciones de Manejo de bits 88
Operaciones de Bifurcación y Subrutinas 88
Operaciones de Transferencia de Bloques y de búsqueda 90
Instrucciones de Control de la CPU y de E/S 91
Modos de Direccionamiento 96

5. LA CONFIGURACION DEL SISTEMA MSX 99

Gestión de la memoria en MSX 99
Acceso a los circuitos de sonido, video y periféricos 100
Introducción general al VDP 100
El circuito de sonido GENERAL INSTRUMENTS AY-3-8919 101
El interfaz programable de periféricos INTEL 8255 101
Control de interrupciones y "Ganchos en RAM" 102
Programa Ejemplo: Reloj de tiempo real 103
Utilización de la RAM en el sistema MSX 107
Utilización de subrutinas en código máquina desde BASIC ... 108

6. EL PROCESADOR DE VIDEO (VDP) 110

Las lineas de control 110
Los registros del VDP 112
Modos de pantalla 116
Modo Gráfico I 116
Modo Gráfico II 117
Modo Multicolor 117
Modo de Texto 119
Sprites (figuras móviles) 119
El procesador de video dentro del sistema MSX 121
Programación del VDP: Consejos y Sugerencias 124
Programa Ejemplo: Definición de caracteres y sprites 124
Definición dinámica de patrones 142
Modo gráfico II, como un sistema de mapa de bits 143
Técnicas de interrupción con sprites 145
Sprites de dos colores 145
Sprites de diferentes tamaños 147
Acceso rápido al VDP 148

7. EL GENERADOR DE SONIDO PROGRAMABLE 150

Los Registros de datos 150
Acceso al generador programable de sonido en el sistema MSX.. 154
Programación del generador de sonido 154
Música en tres canales: El ordenador musical 155
Efectos de sonido en el AY-3-8910 159
Utilización del puerto de sonido de un bit 159

8. OPERACIONES DE ENTRADA/SALIDA 161

Entrada/Salida de mandos de juegos, raquetas y tableros ... 161
Entrada/Salida de teclado y pantalla 162
Selección de ranuras 163
Exploración del teclado: Comprobación de teclas 164

APENDICE A: **Códigos de caracter** 167

APENDICE B: **Códigos de color** 168

APENDICE C: **Tabla de VRAM** 169

APENDICE D: **Instrucciones Z-80** 170

APENDICE E: **TI 9929 A VDP** 173

APENDICE F: **GI AY-3-8910** 186

CAPITULO I

INTRODUCCION

Generalidades. Organización de la Memoria.

Conexión a cassette. Modos de Pantalla.

Circuitos de Video y Sonido

GENERALIDADES

El sistema MSX es una especificación presentada por Microsoft Inc. a mediados de 1.983. Incluye el lenguaje BASIC, sistema operativo y conectores externos (incluyendo puertos para cartuchos de ROM y mandos de juegos). Los programas son intercambiables entre todas las máquinas MSX (siempre que esté disponible la memoria adecuada). Todos los modelos (excepto el Spectravideo SVI-728 y el modelo Yamaha CX5M) llevan la unidad de alimentación incorporada. Aquellas características que no se requieren en el sistema básico, tambien están rígidamente especificadas: puerto para impresora paralelo-centronics, segundo puerto para mando de juegos, conexión RS-232C, y segunda ranura para cartucho.

El sistema operativo de disco (MSX-DOS) usa el mismo formato de disco que el MS-DOS, pero no se ha llegado a un acuerdo sobre el tamaño del disco. El MSX-DOS requiere un mínimo de 64K de RAM (memoria de acceso aleatorio).

La unidad central de proceso es un Z-80A ó equivalente, funcionando a 3.58 MHz.,auxiliado por el procesador de pantalla de video (VDP) TMS 9129 (Europa), de Texas Instruments. El VDP tiene 16K de memoria RAM, especialmente dedicada a video, y maneja hasta 32 sprites (figuras móviles), cada una de un único color. Todos los sprites aparecen delante del plano de fondo.

El VDP tambien genera las interrupciones del sistema. Estas ocurren al completarse cada barrido de la pantalla (aproximadamente cada 1/50 seg.), y se usa para la exploración del teclado.

El Z-80 tiene suficiente capacidad de direccionamiento para acceder a una memoria de 64K. Al conectarse, la ROM del sistema ocupa los primeros 32K y como mínimo, deben existir 8K de RAM de usuario adicionales. En la práctica, el método de gestión de memoria limita la RAM mínima a 16K, si bien, la mayoría de los sistemas incorporan 32 ó 64K.

El intérprete BASIC sólo es capaz de utilizar 32K de RAM, independientemente de que exista RAM adicional, de èstos, 28K están disponibles para almacenamiento de programas. El BASIC MSX es un BASIC Microsoft standard (versión 4.5) con extensiones especiales para la manipulación de gráficos y sonido. Las principales caracteristicas incluyen un editor de pantalla completa, instrucciones dirigidas mediante interrupciones y 14 dígitos de precision con el tipo de variable asumido por omisión.

La capacidad de generación de sonido se obtiene con el generados programable de sonido (PSG) AY-3-8910 de General Instruments, que produce tres canales sobre un espectro de 8 octavas. El PSG tambien maneja la entrada de mando de juegos; la especificación mínima MSX requiere un único puerto standard de tipo D.

Se utiliza un dispositivo de entradas/salidas programable en paralelo, tipo 8255 (PPI), para seleccionar los bloques de memoria. Estos aparecen ante el procesador divididos en cuatro "ranuras" de 16K. Tambien se utiliza para la exploración del teclado y en el caso de que exista un puerto para impresora (extensión standarizada).

Una segunda extensión standarizada opcional es la conexión RS-232C. Los componenetes utilizados en la misma son el circuito de comunicaciones 8251 y el temporizador programable 8253.

Es obligatoria una ranura para cartucho, siendo dos las fijadas en la especificación. En el momento de la impresión se han anunciado los siguientes modelos en el mercado inglés:

Hitachi MB-H80
Toshiba HX-10
Sony HB-75 (48K ROM)
Sanyo MPC 100
JVC HC-7GB
Yashica YC-64
Sega-Yeno DPH-64
Goldstar FC-20
Canon V-20
Yamaha CX5M (sintetizador + ordenador MSX)
Mitsubishi MLF-48
Mitsubishi MLF-80
Panasonic CF2800

ORGANIZACION DE LA MEMORIA

El Z-80 puede leer y escribir datos a un area de memoria de hasta 64K. El diseño del MSX permite que estos 64K sean seleccionados de un banco de memoria mayor. Este banco está compuesto de cuatro "ranuras" primarias, cada una de las cuales puede contener hasta 64K La selección de memoria se hace en bloques o páginas de 16K. El area de memoria direccionable por el Z-80 contiene cuatro páginas de 16K.

El puerto A del PPI 8255 se usa para determinar qué "ranura" provee cada página de 16K. Los dos bits menos significativos especifican la ranura que proveerá la página de 0 á 16K, y así, sucesivamente.

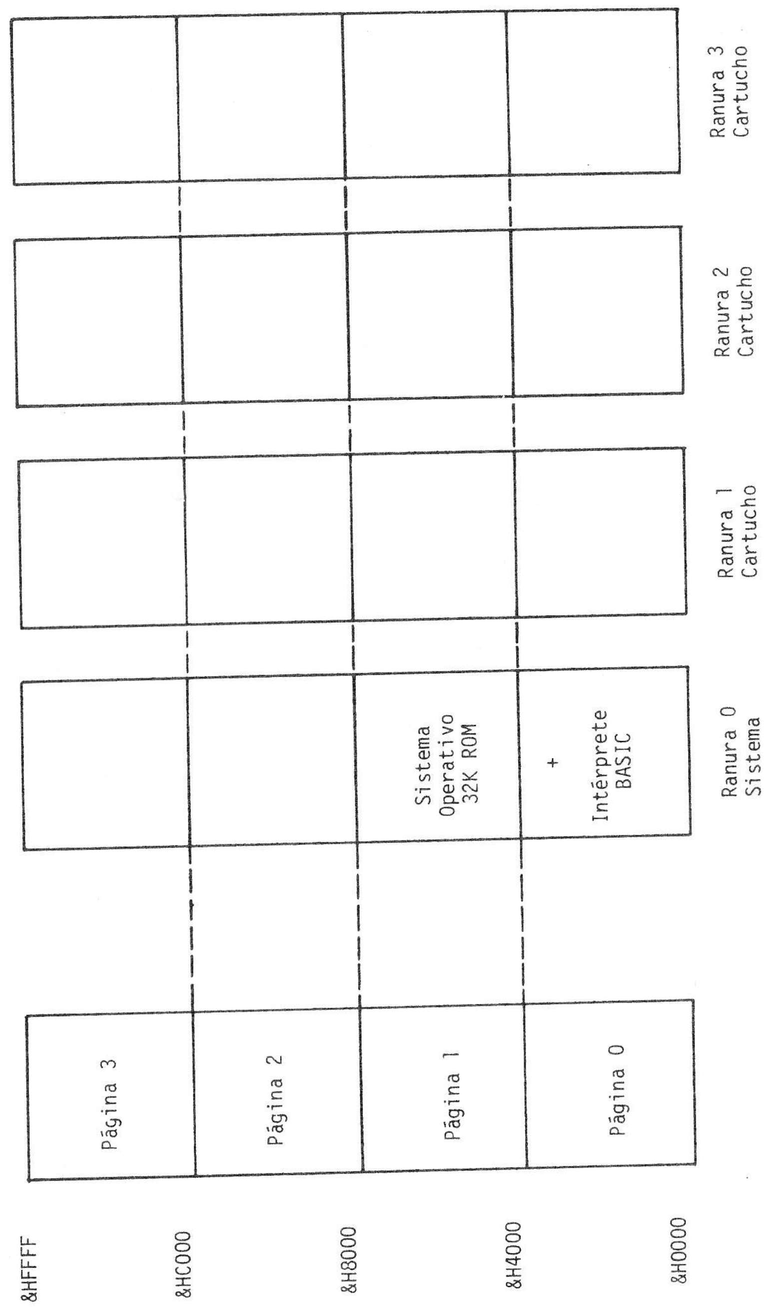


Figura 1.1 Mapa de memoria

Debe reseñarse que una página solamente puede aparecer ante el procesador, en la misma situacion que ocupa en la "ranura" primaria. Es decir, la pagina de la "ranura" cuatro, que ocupa desde los 48K a los 64K solamente aparece ante el procesador en esa posición, no haciéndolo en las que van de 0 á 16K, de 16 á 32K, ni de 32 á 48K.

Las "ranuras" primarias están numeradas de 0 á 3. La "ranura" 0 alberga los 32K de ROM del sistema operativo e intérprete. La mayoria de los ordenadores MSX tienen 64K de RAM, que están situados en una única "ranura". Por ejemplo, el Sanyo MPC-100 utiliza la "ranura" 3 para contener la RAM, mientras que el Toshiba HX-10 utiliza la 2. Los cartuchos usan una de las "ranuras" sobrantes, mientras que las otras pueden configurarse tanto como conexión de cartucho como bus de expansión. El almacenamiento de los programas en BASIC comienza en la posición 32768.

PUERTOS DE ENTRADAS/SALIDAS

Además de poder direccionar 64K de memoria, el Z-80 puede gestionar la entrada o salida a 256 puertos de 8 bits. Los puertos 128 al 255 (&H80-&HFF) están asignados a componentes del sistema y sus extensiones: RS-232C, impresora, procesador de pantalla, generador de sonido, lapiz de luz, etc. El resto están reservados. La interrupción no enmascarable (NMI) no puede ser utilizada, puesto que la dirección del vector 66H se usa por el MSX DOS.

Las E/S deben manejarse utilizando las rutinas residentes en la ROM porque no existe la garantía de que las localizaciones del sistema sean homogéneas entre los distintos aparatos. Debe hacerse una excepción en el caso del procesador de pantalla, cuando sea necesaria una rápida transferencia de datos.

CONEXION A CASSETTE

No se requiere una unidad de cinta especial. La velocidad de transferencia por omisión es de 1200 baud. (aproximadamente bits/seg) y puede operar hasta 2400 baud. La detección y ajuste a la velocidad no usual es automática. La conexión externa se realiza mediante un conector DIN de ocho patillas, en el cual también está previsto el control remoto del motor. La modulación se realiza en frecuencia (FSK) bajo el mando del software. Los niveles óptimos de grabación y reproducción dependen de cada grabadora; sin embargo, casi siempre es necesario situar el volumen próximo al máximo. La instrucción de BASIC, MOTOR, permite controlar el motor de la cinta.

Pueden emplearse tres categorías de archivos:

1. Archivos de programa en cassette. Las instrucciones son CLOAD, CSAVE y CLOAD? para verificación.
2. Archivos ASCII utilizando SAVE y LOAD, no existe instrucción de verificación. Los programas en ASCII pueden fusionarse.
3. Reproducción de memoria, usando las instrucciones BSAVE y BLOAD. Tampoco existe instrucción de verificación.

MODOS DE PANTALLA

El TMS 9129A es capaz de mostrar un plano de fondo con quince colores ademas del transparente, aparte de los sprites (figuras móviles).

Existen cuatro modos de pantalla, dos de texto y dos de gráficos:

1 Modo de texto con 40 columnas y 24 líneas, dos colores y sin posibilidad de utilización de sprites. Los caracteres están formados por una matriz de 8x6 puntos. Se dispone de 256 caracteres diferentes.

2. Modo de texto con 32 columnas y 24 lineas, y 16 colores. Matriz de 8x8 puntos. 256 caracteres diferentes.

3. Modo gráfico de alta resolucion con 32 columnas y 24 lineas, y 16 colores. Similar al modo de texto de 32x24, aunque con un juego de 768 caracteres para permitir que la pantalla se recoja como en un mapa en la memoria (bit-mapped). Adicionalmente, se almacena la información sobre el color.

4. Modo gráfico multicolor con 16 colores, cada bloque de 4x4 puntos puede representarse en un color. No existen caracteres disponibles.

El modo de pantalla se selecciona, desde BASIC, utilizando la instruccion SCREEN, o modificando los tres bits de modo en dos de los ocho registros de escritura del VDP (M3 bit 6 del registro 0, M1 y M2 bits 3 y 4 del registro 1). En todos los modos, excepto en el modo de texto de 40 columnas, el color del borde puede fijarse independientemente.

El VDP explora los 16K de RAM de video. Esta RAM es independiente del área de memoria del Z-80, y puede leerse o escribirse por la CPU (unidad central de proceso) solamente a través del VDP. El VDP no incorpora la posibilidad de hacer un scroll de la pantalla. (Movimento ascendente del contenido de la pantalla).

La instrucción de BASIC que permite borrar la pantalla es CLS. Es un poco extraña, puesto que opera en todos los modos de pantalla, sin distincion.

EL VDP (PROCESADOR DE PANTALLA)

El circuito empleado es el TMS 9129A VDP de Texas Instruments, con cuarenta patillas en encapsulado "dual in line" (DIL). Utiliza 16K de RAM dinámica, a la que solo él puede acceder directamente, utilizando un bus de datos bidireccional de 8 bits El control del VDP se efectúa mediante tres líneas: RAS, CAS y R/W, que proceden del bus de direcciones y de las líneas de control del procesador. El VDP tambien está conectado al bus de datos del sistema.

La lectura o escritura a la RAM de video puede realizarse desde BASIC utilizando las instrucciones VPOKE y VPEEK, ó desde el nivel de lenguaje ensamblador a través de las rutinas en ROM apropiadas.

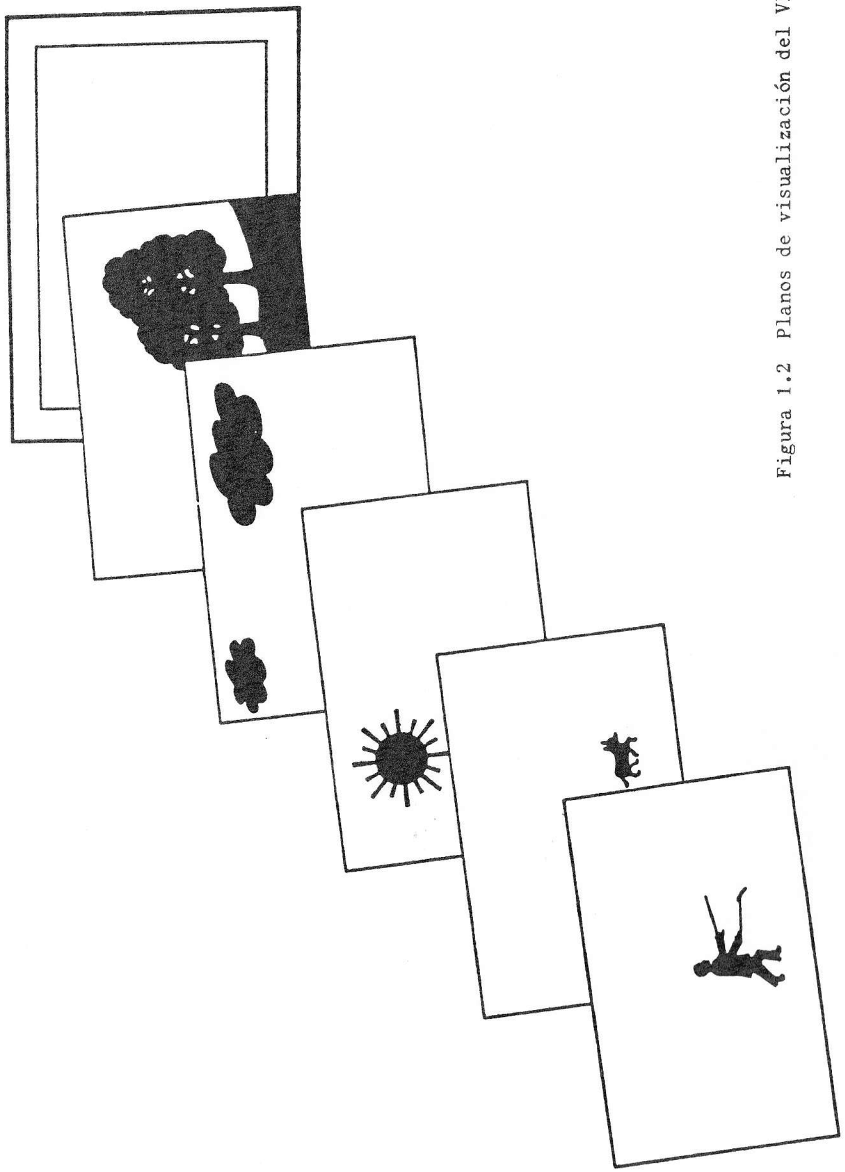


Figura 1.2 Planos de visualización del VDP

ESTRUCTURA DE PANTALLA EN EL VDP

En todos los modos el VDP construye la pantalla fundamentalmente a partir de dos tablas contenidas en la RAM de video:

1. Tabla de patrones generadores.

2. Tabla de nombres de patrones

La tabla de patrones generadores contiene la definición de cada carácter que puede ser mostrado (por ejemplo, la definicion de los puntos de cada carácter). La situación del comienzo de la tabla de generadores en cada modo de pantalla es la siguiente:

Modo 0:	Texto 40:	2048
1:	32:	0
2:	HRG:	0
3:	Multicolor:	0

En todos los modos, excepto en el multicolor, cada carácter está definido en la tabla de patrones generadores utilizando ocho bytes, para especificar el patrón de puntos. Los caracteres standard están justificados a la izquierda, teniendo las dos columnas de la derecha y la fila inferior en blanco, como puede apreciarse en la figura 1.3.

En el modo de texto de 32 columnas pueden usarse hasta 256 definiciones de caracteres; en este caso, la tabla tiene 2048 bytes de longitud. El modo HRG (gráficos de alta resolución) permite definir hasta 768 caracteres, obteniéndose una tabla de 6144 bytes de longitud. El modo de texto de 40 columnas permite 256 definiciones de caracteres, de nuevo cada una de 8 bytes; sin embargo, las dos columnas de la derecha no se muestran. De esta forma pueden obtenerse 40 columnas, puesto que cada carácter está formado por 8x6 puntos en vez de 8x8 puntos, como sucede en el modo de texto de 32 columnas y en el modo HRG

En el modo multicolor, la tabla de patrones generadores contiene 192 elementos, cada uno de 8 bytes, divididos en cuatro pares de bytes Cada par no define un carácter, sino el color de los grupos de 4x4 puntos dentro de un bloque de caracter de 8x8.

El esquema de la tabla de patrones generadores puede verse en la figura 1.4.

Ejemplo: En el modo de texto de 32 columnas, la tabla de generadores empieza en la posición 0 de la VRAM, por lo tanto, el carácter con el código 65 -una "A"- está definida desde la posición 65x8 = 520, hasta la 527.

La tabla de nombres de patrones determina qué carácter va a aparecer en cada posición de carácter (en el modo multicolor los colores están determinados).

En el modo de texto de 33 columnas hay 32 filas de 24 columnas, y, por tanto, tendremos 32x24 = 768 posiciones de caracteres. Cada una de ellas puede contener uno de los 256 caracteres, y en consecuencia, la tabla de nombres tiene 768 bytes de longitud -cada byte especifica el carácter que aparecerá en una posición determinada-.

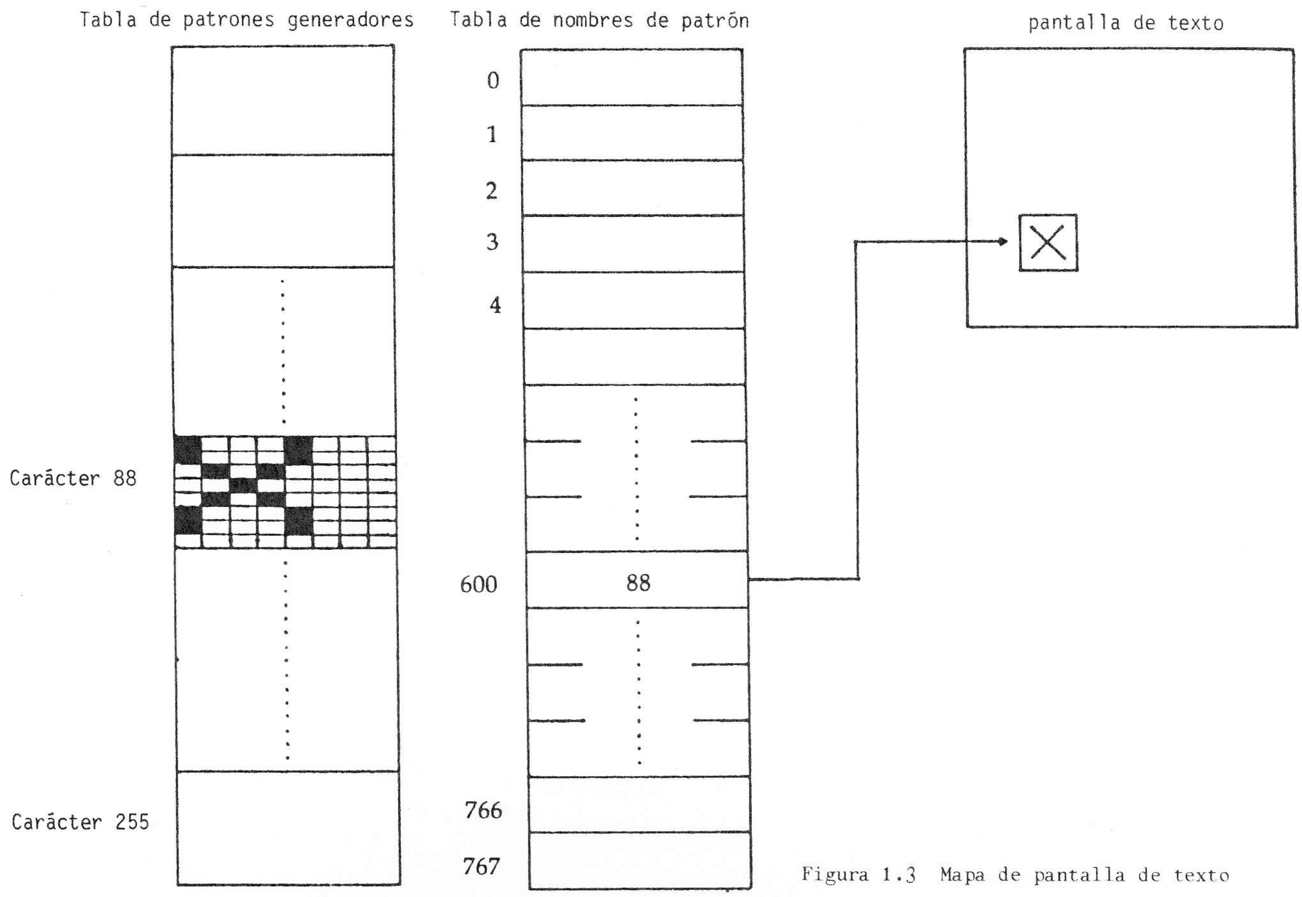


Figura 1.3 Mapa de pantalla de texto

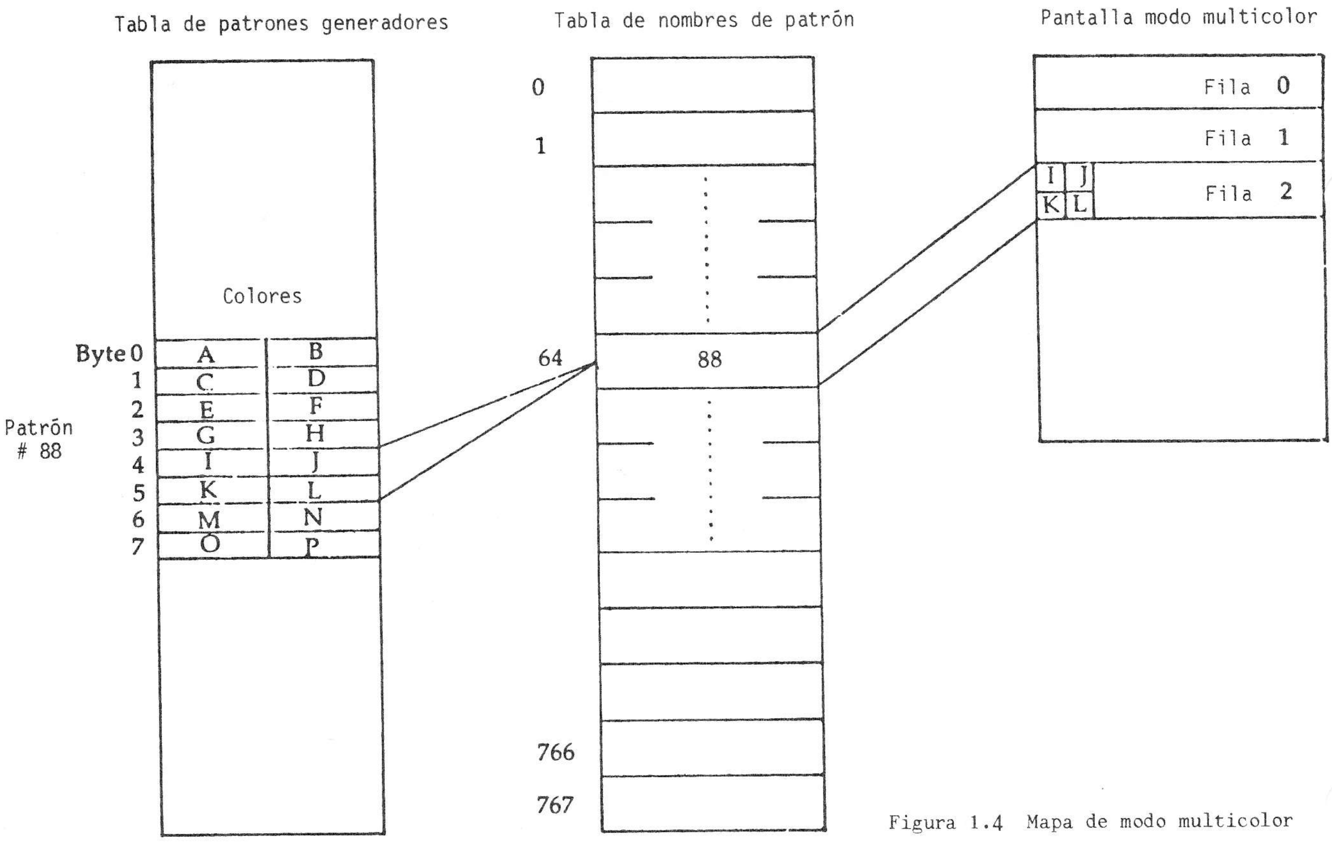


Figura 1.4 Mapa de modo multicolor

En modo de texto de 40 columnas, la tabla de nombres tiene 40x24 =960 bytes de longitud. De nuevo, cada byte especifica el patrón mostrado en una de las posiciones de pantalla. Al igual que en el modo de texto de 32 columnas, la elección se hace de entre 256 variantes contenidas en la tabla de patrones generadores.

GRAFICOS DE ALTA RESOLUCION

En el modo HRG hay, de nuevo, 768 bytes en la tabla de nombres, sin embargo, como en este modo hay 768 definiciones de caracteres disponibles, ¿cómo puede un byte efectuar la selección?

La pantalla se divide en tres zonas, y los primeros 256 elementos en la tabla de nombres seleccionan los caracteres 0-255, los siguientes 256 elementos seleccionan los caracteres 256-511, y los ultimos 256 elementos seleccionan los caracteres 512-768.

La tabla de nombres en modo multicolor tiene 768 bytes de longitud. El valor contenido en cada byte es una referencia a un bloque de 8 bytes de la tabla de generadores.

Dos bytes de este bloque especifican los colores de los cuatro bloques de 4x4 puntos, que conforman el espacio de un carácter. La fila en que se encuentra el caracter (cuatro bloques de 4x4) determina el par de bytes, de los cuatro existentes, que controlan el color. Si el carácter se encuentra en la fila superior, se utilizan los dos primeros bytes, si está en la segunda, los bytes tres y cuatro, y así, sucesivamente. Por lo tanto, las filas 0, 4, 8, 12, 16 y 20, corresponden a los dos primeros bytes de cada bloque.

COLOR

Aunque el VDP puede generar hasta quince colores diferentes, además del transparente (incluimos el blanco y negro como colores), en el modo de texto de 40 columnas solo pueden utilizarse dos colores. El borde toma el color del fondo.

Desde BASIC, se cambian los colores (como en cualquier otro modo) utilizando la instrucción COLOR, mientras que en lenguaje máquina, el registro 7 de escritura del VDP determina ambos colores. Este registro se altera más fácilmente utilizando la rutina apropiada en ROM.

En el modo de texto de 32 columnas, la tabla de patrones generadores de 2K está dividida en grupos de definiciones de ocho caracteres. Para cada grupo, el color de fondo y de primer plano, se determina mediante un byte de la tabla de generación de color. Por lo tanto, la tabla de color tiene 32 bytes de longitud.

El color de primer plano se encuentra en el nibble (grupo de cuatro bits) más significativo, y el color de fondo, en el nibble de orden más bajo. Si queremos mostrar una letra dos veces, pero en colores distintos, la definición del caracter debe duplicarse en otra posición de la tabla de patrones. De esta forma, puede asignarse un color diferente. En el modo HRG la combinación color de primer plano/fondo, puede especificarse para cada una de las ocho lineas de una posición de

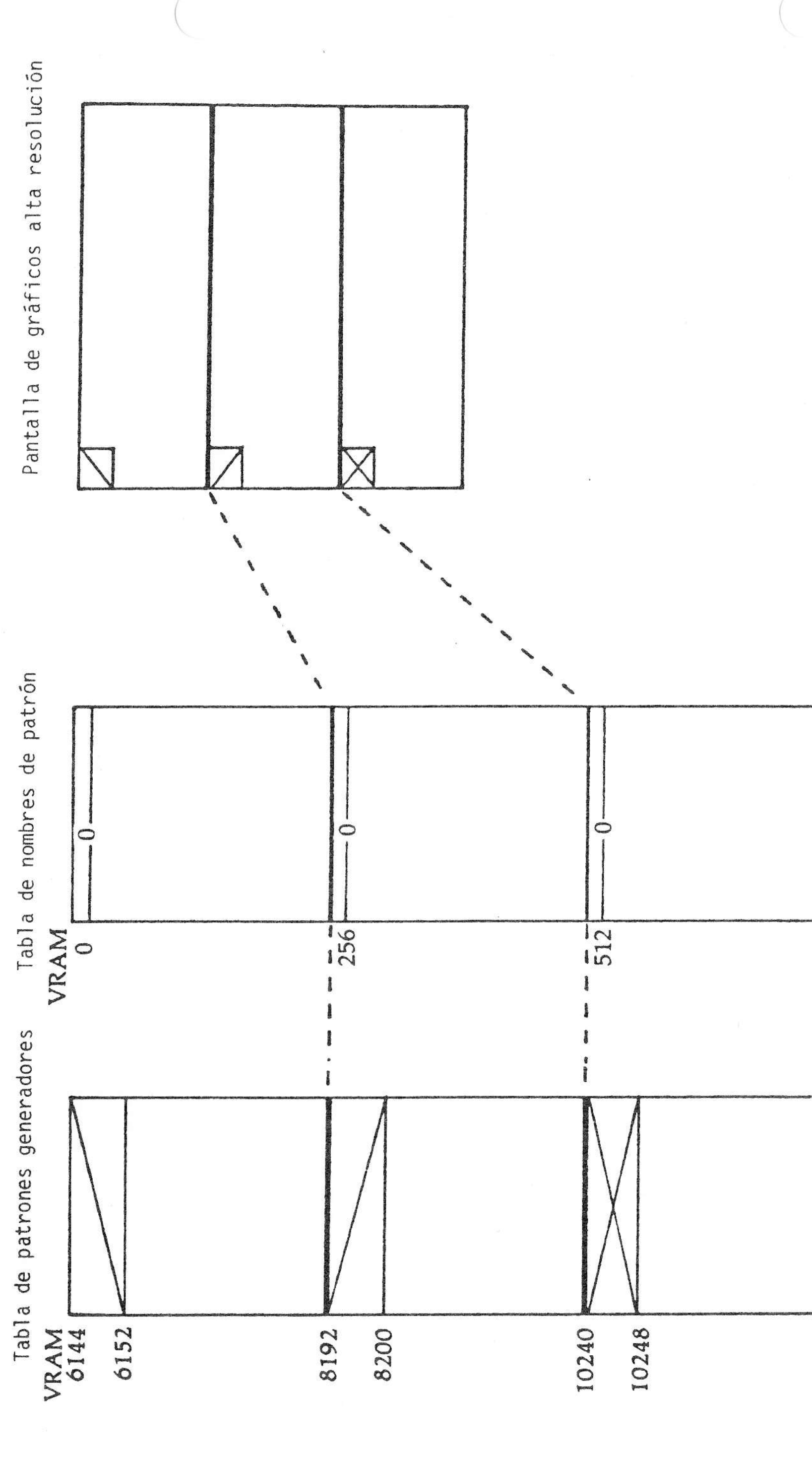


Figura 1.5 Modo de gráficos alta resolución

carácter. Se requieren 768x8 = 6144 bytes, que se organizan en el mismo formato que la tabla de patrones generadores, tambien de 6144 bytes. Este método de asignación de color hace que la realización de una rutina de scroll sea muy compleja.

SPRITES (FIGURAS MOVILES)

Un sprite es un patrón que puede ocupar la posición de 1 caracter, de 2x2 ó de 4x4 caracteres. Es independiente del plano de fondo, y se mueve alterando las coordenadas X,Y asignadas al sprite. Puede mostrarse o desaparecer a voluntad, y además, el choque de sprites se detecta automáticamente. Cada sprite tiene una cierta prioridad y si un sprite de superior prioridad, ocupa la misma posición que otro, se superpone a éste, ocultándolo, excepto en las zonas transparentes del primero. Los sprites permiten realizar efectos especiales; por ejemplo, simulaciones tridimensionales.

Los sprites no pueden utilizarse en el modo de texto de 40 columnas. En todos los otros modos pueden disponerse hasta 32 sprites en pantalla, con un máximo de 4 por línea. Si se excede de este número, solamente se verán los cuatro sprites de mayor prioridad en esa línea. En ese momento, la bandera (flag) perteneciente a este quinto sprite, toma el valor 1, y el número del quinto sprite se almacena en el registro de estado del VDP.

Cada sprite puede tomar un único color, y su prioridad no puede alterarse. El control de los sprites se realiza desde BASIC mediante dos instrucciones. La primera, SPRITE$ (n), se utiliza para definir un grupo se patrones de sprite, y la segunda, PUT SPRITE, fija la posición, color y número de patrón de cada sprite mostrado.

Estas instrucciones modifican dos tablas de la RAM de video, que contienen todos los datos de los sprites. La tabla de patrones de sprite, que comienza desde la posición 14336 de la VRAM -independientemente del modo de pantalla-, contiene los datos de la forma de todos los sprites. La tabla de atributos de sprite almacena las coordenadas X,Y, el color y el patrón de cada sprite.

Los sprites pueden formarse por 8x8 ó 16x16 puntos. Además, pueden mostrarse al doble de su tamaño normal -de esta forma, el mayor sprite posible tendrá 32x32 puntos-. Los sprites se sitúan en pantalla con respecto a la esquina superior izquierda, y pueden moverse de punto en punto.

La tabla de patrones de sprite permite acomodar hasta 256 bloques de 8 bytes. Si los sprites van a ser de 8x8 puntos (no pueden mezclarse sprites de 8 y 16 puntos a la vez), entonces pueden definirse 256 patrones de sprite independientes. El patrón cero ocupará las posiciones 14336-14343, el patrón uno las comprendidas entre 14344-14351, etc. Si se selecciones los sprites de 16 puntos, habrá que definirlos mediante cuatro bloques consecutivos de 8 bytes. Cada bloque definirá una cuarta parte del sprite según la siguiente secuencia:

1	3
2	4

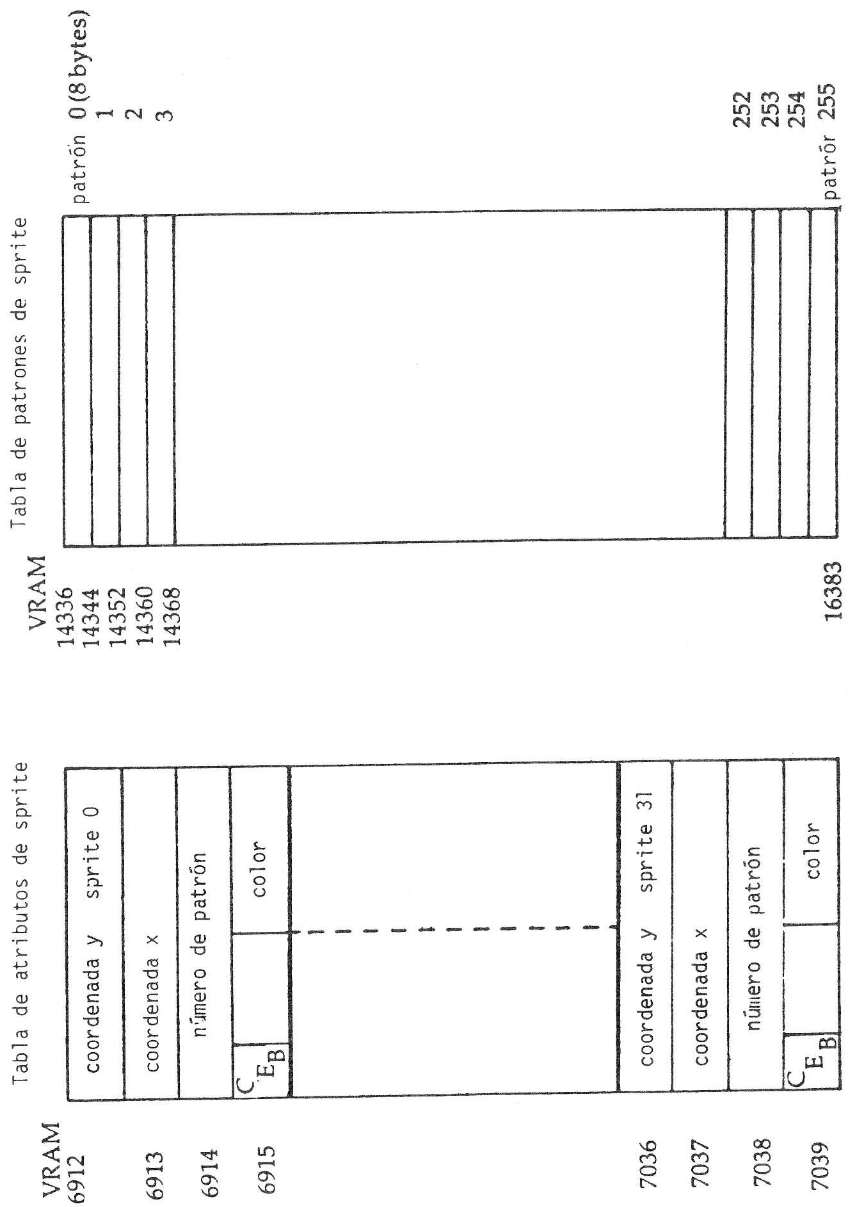


Figura 1.6 Tablas de sprite

Después de fijar la forma de los sprites, se situan en pantalla mediante un elemento de la tabla de atributos de sprite. En todos los modos de pantalla -con capacidad de usar sprites- la tabla empieza en la misma posición de la VRAM, la 6912, y tiene una longitud máxima de 32 sprites x 4 bytes = 128 bytes. Cada elemento especifica la posición, color y número de patrón de cada sprite. Los primeros 2 bytes contienen las coordenadas Y,X, y el tercero identifica la forma seleccionando un bloque o bloques de la tabla de patrones de sprite. El byte final contiene en su nibble, de orden inferior, el color del sprite. Además el bit más significativo "bit previo al reloj", si se encuentra a nivel 1, desplazará la posición del sprite 32 puntos hacia la izquierda.

Las colisiones entre sprites son utilizados, desde BASIC, mediante las instrucciones SPRITE ON y ON SPRITE.

La figura 1.2 muestra la combinación de planos de pantalla producida por el VDP.

CIRCUITO DE SONIDO

El circuito empleado es el AY-3-8910 PSG (generador programable de sonido). Se trata de un generador de sonido programable mediante registros, capaz de producir tres canales de sonido sobre ocho octavas. Tambien controla la entrada del mando de juegos, mediante dos puertos de E/S independientes.

Cada canal de sonido tiene un control separado de volumen, y puede producir tono y/o ruido. Sin embargo, la misma frecuencia de sonido y el mismo ciclo y forma de envolvente, se emplea en todos los canales. Si dos canales producen ruido, generarán el mismo ruido, sólo la amplitud podrá variar.

El PSG tiene 16 registros internos de lectura/escritura, dos de los cuales funcionan como registros de almacenamiento para los dos puertos de los dos mandos de juegos. Los restantes registros pueden dividirse en cinco grupos en base a su función:

1. R0-R5: Control de frecuencia de canal.

2. R6: Generador de frecuencia de ruido.

3. R7: Registro de selección de canal de tono y/o ruido. Los dos bits más significativos fijan el sentido de transferencia de datos de los dos puertos de E/S.

4. R10-R12 (Octal): Registro de selección de constancia de tono o de control por envolvente.

5. R13-R15 (Octal): Registro de selección del patrón de envolvente y período.

Los dos puertos de E/S son completamente independientes de las funciones de generación de sonido del PSG.

El BASIC MSX incluye un extenso grupo de instrucciones musicales, para las cuales la mayoría de los parámetros están predeterminados. En combinación con la capacidad de temporización de interrupciones, es posible obtener complejos efectos de sonido.

CAPITULO 2

BASIC MSX

Generalidades. Variables y Funciones.

Instrucciones Gráficas. Sonido.

Almacenamiento de Programas.

GENERALIDADES

El BASIC MSX es un BASIC Microsoft standard, versión 4.5, con varias extensiones potentes especialmente en gráficos y sonido. Las omisiones principales incluyen los procedimientos, las formas WHILE/WEND y REPEAT/UNTIL y abreviaturas de las palabras clave.

Incluye un editor de pantalla completa que permite corregir fácilmente cualquier línea en pantalla. La reentrada de línea ocurre al pulsar la tecla RETURN, estando el cursor sobre cualquier caracter de la línea. No se produce comprobación de errores durante la entrada.

Una línea puede contener hasta 255 caracteres, incluyendo sentencias múltiples. En las líneas con sentencias múltiples, se emplea el signo ":" como separador. Una sentencia REM pasa la ejecución del programa a la siguiente línea. Los números de línea deben estar comprendidos entre 0-65529, inclusive.

Los espacios no son necesarios, y son ignorados por el intérprete (salvo que formen parte de una cadena). En consecuencia, las palabras clave no pueden ser incluidas dentro de nombres de variable (solamente los dos primeros caracteres son reconocidos). Las mayúsculas y minúsculas se consideran diferentes. Las palabras clave son aceptadas, tanto en mayúsculas, como en minúsculas.

Además del teclado principal, hay dos grupos de teclas. A la derecha encontramos el grupo de edición -las teclas de cursor y las opciones de inserción y borrado-. A la izquierda existe un grupo de cinco teclas de función. Al conectar el ordenador, las teclas de función quedan asignadas con diez funciones. Cinco de ellas se obtienen directamente y las otras cinco, pulsando las teclas de función en combinación con la tecla SHIFT. El primer juego aparece en la línea inferior de la pantalla -esta línea puede borrarse, aunque manteniendo el contenido de las teclas, utilizando la instrucción KEY OFF-.

Para redefinir la función de cualquier tecla, se emplea la instrucción KEY x,"cadena". Por ejemplo, para hacer que la tecla 1 ejecute un programa, puede utilizarse la siguiente sentencia: KEY1, "RUN"+ CHR$(13).

istado o ejecución de programa puede detenerse sando la tecla STOP, una vez. Al volverla a pulsar, continuará la acción. Para finalizar ambos, deben pulsarse, simultáneamente, las teclas CONTROL y STOP.

Para aquellos que no estén familiarizados con el BASIC Microsoft, deben destacarse algunas de sus características:

1. La detección de errores se simplifica mediante el uso de TRON y TROFF. Si la sentencia TRON se utiliza dentro de un programa, o directamente, cada vez que una línea es ejecutada, se imprime su número de línea. Este proceso, normalmente, llena la pantalla de números.

2. Pueden borrarse bloques de líneas utilizando la instrucción DELETEx,y y renumerarse mediante la instrucción RENUMx,y,I. También permite la comprobación de desviaciones a líneas inexistentes. En ese caso, en el momento de la ejecución el BASIC MSX no señalará error. Por el contrario, se efectuará una desviación en la ejecución hacia la primera línea que tenga un número mayor que el especificado.

3. Todas las variables que comiencen con un caracter o grupo de caracteres determinado, pueden declararse de un tipo determinado:

Entero: DEFINT, precisión sencilla: DEFSNG, precisión doble: DEFDBL ó cadena: DEFSTR.

4. Los conjuntos pueden borrarse mediante la instrucción ERASE.

5. Al conectar el ordenador, el espacio asignado para almacenamiento de cadenas es de 200 bytes. Una asignación distinta precisa el uso de la instrucción CLEAR. Por ejemplo, para reservar 1000 bytes, hay que utilizar CLEAR 1000.

6. La función FRE(0) nos entrega la cantidad de memoria restante para almacenamiento de programa.

7. La función de cadena MID$ permite reemplazar un caracter o varios caracteres, dentro de una misma expresión.

8. La estructura GOSUB/RETURN permite que la dirección de retorno quede especificada: RETURN número de línea.

VARIABLES

Los nombres de variable pueden tener cualquier longitud, pero deben de empezar con una letra. Sólo los dos primeros caracteres son significativos. Las letras minúsculas se distinguen de las mayúsculas. Las variables no precisan ser asignadas con un valor inicial, asumen un valor nulo si ninguno ha sido declarado. Tampoco necesitan ser dimensionadas, a no ser que el número de elementos exceda de once, por ejemplo, S(0) a S(10) no precisan ser dimensionadas.

Cuando una variable no ha sido declarada de un tipo particular, asume ser una variable numérica, real, de doble precisión, y es almacenada con catorce dígitos de precisión. Las variables declaradas como de precisión sencillas se almacenan con seis dígitos de precisión. Los enteros están comprendidos entre -32768 y 32767.

Una variable precisión sencilla se declara añadiéndole un si de admiración "!", por ejemplo, SP!. El tipo entero con el signo de porcentaje "%". Si es preciso declarar una variable como de doble precisión, se utiliza el signo numérico "#".

Como ya se ha dicho anteriormente, las variantes de la sentencia DEF, se emplean para definir globalmente tipos de variable. Si la sentencia DEFINT A se ha utilizado, todas las variables que comiencen por "A" serán consideradas enteras. En este caso, la variable "A" no será diferenciada de la "A%".

Dimensionar variables es sencillo, por ejemplo, DIMX(20,40),Y(80,160). Sin embargo, hay un límite de 255 dimensiones, aunque el número de elementos está limitado solamente por las necesidades de memoria.

Como quiera que cualquier variable no necesita ser dimensionada, a menos que se precisen más de once elementos, puede conseguirse un ahorro de memoria si dichas variables de menos de once elementos se dimensionan.

El BASIC MSX incluye la instrucción SWAPx,y, que intercambia valores entre las variables. Ambas variables deben ser del mismo tipo. Otra instrucción interesante es VARPTRx, que devuelve la dirección de almacenamiento de la variable especificada, si es que ha sido declarada.

Las variables del sistema MSX son:

1. TIME: es el reloj del sistema, incrementado cada 1/50 seg.

2. BASE(n): devuelve la primera posición en la RAM de video de la tabla especificada, independientemente del modo de pantalla.

3. VDP(n): devuelve el valor del registro de escritura del VDP especificado o el registro de estado.

4. SPRITE$(patrón #): utilizado para definir cada una de las 256 formas de sprite de 8x8 puntos, o de las 64 de 16x16 puntos. Debe igualarse a una cadena, como máximo, de 32 caracteres. El código de cada carácter fija el patrón de bits para un byte de definición.

FUNCIONES

Están incluidas todas las ya conocidas (XYZ es un valor arbitrario):

ASC(XYZ$), CHR$(XYZ), LEN(XYZ$), INT(XYZ), EXP(XYZ), ABS(XYZ), LOG(XYZ), HEX$(XYZ), OCT$(XYZ), BIN$(XYZ), SGN(XYZ), RND(XYZ), MID$(XYZ$,N,n), RIGHT$(XYZ$,n), LEFT$(XYZ$,n), STR$(XYZ), VAL(XYZ$), STRING$(n,XYZ$), INSTR(XYZ$,xyz$), ATN(XYZ), COS(XYZ), SIN(XYZ), SQR(XYZ), TAN(XYZ), TAB(n), SPC(n), INKEY$, INPUT$(n), PEEK(XYZ), POKEXYZ,n, FRE(0), FRE(""), USRXYZ(n), VARPTR(XYZ), VARPTR(#XYZ), CINT(XYZ), POS(n) y LPOS(n).

Las extensiones incluyen:

1. CDBL(XYZ) y CSNG(XYZ) que convierten XYZ a precisión doble y sencilla, respectivamente.

2. VPEEK(n) y VPOKEn,x son los equivalentes de las instrucciones PEEK y POKE para la RAM de video.

3. STICK(n) y STRIG(n) permiten obtener la dirección de un mando de juegos o el accionamiento del disparador (tambien actúan con las teclas de cursor y la barra de espacio).

4. POINT(X,Y) devuelve el color del punto especificado.

5. PLAY(canal) comprueba si se está sonando música en alguno de los canales.

6. EOF(archivo #) comprueba si se ha alcanzado el final del archivo secuencial.

7. PAD(n) devuelve una serie de valores procedentes de un tablero digital, en función del parámetro "n".

8. PDL(pala #) retorna el valor procedente de una pala de juegos.

INSTRUCCIONES GRAFICAS

Se dispone de una paleta completa de quince colores, además del transparente, en todos los modos (veáse apéndice B). Las instrucciones gráficas pueden dividirse en tres apartados:

1. Configuración general y color.

2. Sprites (figuras móviles).

3. Alta resolución.

INSTRUCCIONES GENERALES

El modo de pantalla se fija utilizando la instrucción SCREEN:

SCREEN 0:	modo de texto de 40x24
SCREEN 1:	modo de texto de 32x24
SCREEN 2:	modo gráfico de alta resolución (HRG)
SCREEN 3:	modo multicolor

Esta instrucción tambien se utiliza para determinar el tamaño de sprites, para la conexión/desconexión del eco de pulsación de teclas, para fijar la velocidad de transferencia del cassette y la opción de impresora. Cuando se emplea, para seleccionar un modo de texto, el juego de caracteres se copia desde la ROM, en la VRAM. Esta instrucción no deberá emplearse si el usuario necesita acceder a caracteres redefinidos.

MODOS DE TEXTO

La anchura de las pantallas de texto, tanto de 40 como de 32 columnas, puede fijarse a cualquier valor, a partir de una columna, utilizando la

instrucción WIDTH, por ejemplo, WIDTH20. Hay que mencionar que en ambos modos de texto, las dos columnas de la izquierda y la de la derecha, no se utilizan. La mayoría de los receptores de televisión, no alcanzan a mostrar la columna situada más a la izquierda.

En cualquiera de los modos los caracteres pueden situarse en las columnas reservadas introduciendo los valores apropiados en la tabla de nombres de patrones en la VRAM. Por ejemplo, para colocar una "A" en las dos columnas de la izquierda de la fila superior, en el modo de texto, ejecutaremos VPOKE0,65 y VPOKE1,65.

Los colores globales, en cualquiera de los modos, se seleccionan con la instrucción COLOR. Su formato es:

COLOR primer plano, fondo, borde.

El valor asumido por omisión es 15,4,4. En el modo de texto de 40 columnas el color de borde no puede especificarse independientemente, y toma, el color de fondo. En los modos 1, 2 ó 3, el cambio a un fondo negro con borde en gris se consigue indicando: COLOR14,1,1.

En los modos gráficos, el nuevo color de fondo se produce solamente después de una instrucción CLS.

En el modo de texto de 40 columnas, el VDP no permite utilizar ningún otro color. Tampoco pueden emplearse sprites. En el modo de texto de 32 columnas, el VDP puede mostrar textos en colores diferentes de los fijados mediante la instrucción COLOR. Sin embargo, no existen instrucciones específicas en BASIC, para este propósito. Por lo tanto, para conseguir más de un color de primer plano o de fondo en pantalla, se hace preciso realizar cambios en la tabla de color de la VRAM. Esto puede hacerse utilizando la instrucción VPOKE. Una operación parecida debe hacerse si es necesario redefinir caracteres.

En ambos modos de texto, la posición del cursor se determina con la instrucción LOCATE X,Y. El modo de texto de 40 columnas permite imprimir hasta 37 caracteres en cada línea. Aunque se admiten posiciones de columna con un límite superior de 255, solamente aquéllas comprendidas entre 0 y 36, son consideradas. Para el modo de texto de 32 columnas, estas posiciones deberán estar dentro del intervalo 0-28

Una altérnativa a LOCATE 0,0 es PRINT CHR$(11). La instrucción LOCATE también controla el cursor de pantalla.

LOCATE 2,4,0 hace desaparecer el cursor.

LOCATE 2,4,1 hace aparecer el cursor.

La instrucción PRINT, que puede escribirse como "?", utiliza caracteres de formato standard:

1. El punto y coma ";"produce la omisión del caracter de retorno -CHR$(13)- después del último elemento a imprimir especificado. De esta manera, la posición de impresión no se desplaza a la primera columna de la nueva línea.

2. Una coma "," mueve la posición de impresión al comienzo de la siguiente zona de tabulación. Cada zona de éstas tiene catorce columnas de anchura.

3. El signo "+" se emplea para concatenar cadenas. Por ejemplo, A$ = B$ + "XYZ"

TAB(n) y SPC(n) siguen a la instrucción PRINT. El delimitador de cadenas (") puede omitirse al final de la cadena. Así, PRINTSPC(4);"NOMBRE

La variante PRINT USING permite imprimir tablas de números o cadenas en un formato específico. Los cuatro caracteres de control principales son:

1. El signo de admiración (!): produce la impresión únicamente del primer caracter de la cadena. Por ejemplo, PRINT USING "!";"NUEVO" da como resultado la letra "N".

2. La barra inversa ( ): se emplea en la forma "   " para especificar el número de caracteres que van a imprimirse. Este número es igual a dos más el número de espacios entre los dos signos. Por ejemplo, PRINT USING "   ";"NUEVO" produce como resultado "NUE". Si la longitud del campo excede a la de la cadena, se completa con espacios.

3. El signo (&): inserta una cadena determinada en el lugar de un signo & situado en otra cadena. Por ejemplo, Q$="NUEVO":?USING "ALGO &"; Q$ permite obtener las palabras ALGO NUEVO.

Si se define más de una subcadena, esta secuencia se repite para todas ellas. Por ejemplo, Q$="NUEVO":Z$="VIEJO":?USING "ALGO &";Q$,Z$ se imprimirá como ALGO NUEVOALGO VIEJO.

4. El signo numérico (#): se emplea para imprimir los valores numéricos con un cierto número de dígitos, antes y después del punto decimal. Si el valor no tiene la suficiente longitud, se completa con ceros. Por ejemplo, ?USING "###.##";2,4.684 se escribirá como 2.00 4.68

Si el campo es demasiado estrecho para los datos, se obtendrá un signo % delante del valor o se redondeará. Veamos dos ejemplos, ?USING "##";224444 dá %224444 y ?USING "##.##";22.1234 dá 22.12

Si el campo es excesivamente grande para los datos, el valor será justificado a la derecha, o será completado con ceros. Por ejemplo, ?USING "##.##";22.2 dá 22.20 y ?USING "####";22 dá   22

El valor numérico no debe de tener más de 24 dígitos de longitud, en caso contrario, se producirá un error.

El signo numérico (#) se utiliza en combinación con otros caracteres de control, como son: **, ££, coma, +, - y ^^^^

Un signo más (+) a la izquierda o derecha del símbolo numérico (#) hará que el signo del número se imprima delante o a la derecha del mismo, respectivamente.

Asímismo, un signo negativo detrás del último numérico, imprimirá un signo negativo después del valor numérico, si éste tiene argumento negativo. Por ejemplo, ?USING "##-";-2 imprimirá 2-

El doble asterisco (**) situado a la izquierda de los signos numéricos, rellena los espacios iniciales con asteriscos. Por ejemplo,

?USING "**#.#";4.2 resulta **4.2 y ?USING "**##.#";4.2 dá ***4.2

El signo de doble libra (££) imprime un único signo de libra delante del valor. Este caracter de control no puede utilizarse en unión del carácter (^)

Si se sitúa una coma a la izquierda del punto decimal, el número se imprime con una coma a la izquierda de cada tercer dígito. Por ejemplo, ?USING "####,.#';2222.2 dá 2,222.2

Los cuatro signos (^) permiten imprimir los valores numéricos en forma exponencial. De esta forma, queda suficiente espacio para la letra "E", el signo y los dos dígitos del exponente. Por ejemplo, ?USING "##.#^^^^";22.4 resulta 2.2E+01

La pantalla puede borrarse utilizando la instrucción CLS, en todos los modos de pantalla. En los dos modos de texto, una alternativa a CLS es PRINT CHR$(12). La coordenada Y del cursor puede obtenerse de CSRLIN y la coordenada X de POS(x), donde x es un argumento fantasma, pudiendose utilizar con cualquier valor.

MANIPULACION DE LA RAM DE VIDEO

Para obtener caracteres en colores diferentes de los que se han fijado por la instrucción COLOR en el modo de texto de 32 columnas, deben sustituirse los bytes apropiados de la tabla de color de la VRAM con la nueva combinación de color de primer plano/color de fondo.

Como ya se mencionó, cada uno de los 32 bytes de la tabla de color determina el color de un juego de ocho caracteres del total de 256 disponibles. Por lo tanto, para imprimir texto en colores diferentes de los asumidos por omisión, es necesario copiar parte del juego de caracteres en otro lugar de la tabla generadora de patrones.

TABLA DE PATRONES GENERADORES: 0-2047
TABLA DE COLOR: 8192-8223
TABLA DE ATRIBUTOS DE SPRITE: 6912-7039
TABLA DE NOMBRES: 6144-6911
TABLA DE PATRONES DE SPRITE: 14336-16383

Tabla 2.1 Direcciones de la VRAM en el modo de texto de 32 columnas.

Los caracteres gráficos ocupan los 32 primeros bloques de definiciones en la tabla de patrones. Esto obedece al hecho de que los 32 primeros códigos de carácter no se pueden imprimir. Los caracteres gráficos pueden mostrarse en pantalla mediante: PRINT CHR$(1)+CHR$(65) hasta PRINT CHR$(1)+CHR(95). El carácter con código 255 tampoco es standard. Este caracter se imprime como el inverso del caracter cubierto por el cursor. Se actualiza cada 1/50 seg.

EJEMPLO: TEXTO EN VARIOS COLORES

Para conseguir un texto en letras mayúsculas con color de primer plano amarillo y color de fondo negro, hay que seguir los siguientes pasos:

Copiar las definiciones de los 26 caracteres a otra posición en la tabla de patrones generadores. El juego de caracteres se define utilizando ocho bytes por cada carácter. Los caracteres comprendidos entre el 65 y el 90 siguen la secuencia recogida en el apéndice B.

En el modo de texto de 32 columnas, la tabla de patrones se extiende desde la posición 0 a la 2048 de la VRAM. Como cada carácter necesita ocho bytes para su definición, el primer byte a copiar se encuentra en la posición: ASC("A")*8=520 y el último byte del carácter final está en ASC("Z")*8+7=727

Los nuevos caracteres en mayúsculas se superpondrán sobre una parte del juego de caracteres existente. Reemplazaremos los caracteres 175-170 que están definidos desde las posiciones 145*8=1160 a la 170*8+7=1367, mediante las siguientes instrucciones:

```
10 SCREEN 1:FOR X=0 TO 207: VPOKE 1160+X,VPEEK(520+X): NEXT
```

Los cuatro bytes de la tabla de color que determinan los colores tomados por los 32 caracteres comprendidos entrre 144 y 176 se encuentran en las posiciones 8210 á 8213. El valor de los cuatro bits más significativos fija el color de primer plano. Los cuatro bits menos significativos determinan el color de fondo.

COLOR DE PRIMER PLANO	COLOR DE FONDO	
		= 10*16 + 1 = 161
Amarillo: 10	Negro: 1	

Tabla 2.2 Entrada en la tabla de color.

Por lo tanto, la siguiente línea de la rutina será:

```
20 FOR X=0 TO 3: VPOKE X+8210,161: NEXT
```

Si imprimimos CHR$(145) aparecerá una "A" amarilla sobre un fondo negro. Si se utiliza la instrucción COLOR, los 32 bytes de la tabla de color, se asignan con los colores especificados. Para obtener, de nuevo, los colores alternativos, debe repetirse la ejecución de la línea 20.

Los caracteres se justifican a la izquierda, por lo tanto, cuando se imprimen con un color de fondo que contrasta con el color de la pantalla pueden resultar difíciles de distinguir.

Este método de determinación de colores de pantalla, aunque un poco complicado, permite realizar efectos especiales con facilidad. Por ejemplo, una pantalla que muestre predominante entre cuatro y ocho caracteres, puede ser animada cambiando dos bytes de la tabla de color.

El siguiente programa permite rediseñar, completamente, el juego de caracteres, y guardarlo o cargarlo en casette. Las teclas de cursor se utilizan para moverse alrededor de la matriz de puntos, y la barra de espacio permite fijar la posición de los distintos puntos.

PROGRAMA EJEMPLO: JUEGO DE CARACTERES

```
10 '******************************
20 '****  JUEGO DE CARACTERES  ****
30 '****  CREACION, GRABACION  ****
40 '****        Y COPIA       ****
50 '******************************
60 '
70 'UTILIZA LAS TECLAS DE CURSOR PARA          MOVERTE POR LA MATR
IZ.
80 'PARA MARCAR EL PUNTO CORRECTO,             PULSA LA BARRA DE E
SPACIO.
90 'LA INSTRUCCION SCREEN PONE A               CERO LA RAM DE VIDE
O.
100 '
110 FOR X=0TO12:READA$,B$:POKE38000!+X,VAL("&H"+A$):POKE3820
0!+X,VAL("&H"+B$):NEXT
120 DEFUSR=38000!:DEFUSR2=38200!
130 DATA 21,21,00,40,00,9C,11,11,40,00,9C,00,01,01,00,00,08,
08,CD,CD,59,5C,00,00,C9,C9
140 ON KEY GOSUB 240,270,290,430:KEY(1) ON:KEY(2) ON:KEY(3)
ON:KEY(4) ON
150 ON STRIG GOSUB 400:STRIG(0) ON
160 KEY OFF:COLOR 14,1,1:SCREEN 1,0:CLS
170 LOCATE0,1:PRINT"F1..GUARDAR  F3..CAMBIAR CHR$":LOCATE0,3
:PRINT"F2..CARGAR   F4..COPIAR CHR$"
180 FORX=0TO7:VPOKE14336+X,VPEEK(224+X):NEXT:PUTSPRITE 0,(14
4,47),8,0
190 FORX=373TO375:VPOKEX,0:NEXT:VPOKE371,24:VPOKE372,24
200 A$=STRING$(8,46):FORX=0 TO7:LOCATE16,X+6:PRINTA$:NEXT
210 GOSUB 290:X1=1:Y1=1
220 X2=STICK(0):IF X2=0 THEN 220
230 ON X2 GOTO 350,220,360,220,370,220,380
240 Z=USR(2):LOCATE0,20:PRINT"PREPARA LA CINTA PARA GRABAR Y
 PULSA RETURN"
250 A$=INPUT$(1):IF ASC(A$)<>13THEN 250 ELSE BSAVE"CAS:",400
00!,42047!
260 LOCATE 0,20:PRINTSTRING$(60,32):RETURN
270 LOCATE2,20:PRINT"CARGANDO..":BLOAD"CAS:":Z=USR2(2)
280 LOCATE 2,20:PRINTSTRING$(28,32):RETURN
290 LOCATE0,20:PRINT"INTRODUCE NO. DE CHR$ Y PULSARETURN
   ":GOSUB 470:CN=XX
300 FOR X=0TO7:A$(X)=BIN$(VPEEK(CN*8+X)):NEXT
```

```
310 FORX=0TO7:IF LEN(A$(X))<8 THEN A$(X)=STRING$(8-LEN(A$(X)
),79)+A$(X)
320 FOR BT=1TO8:LOCATE15+BT,X+6:IF MID$(A$(X),BT,1)="1" THEN
 PRINTCHR$(219) ELSE PRINTCHR$(46)
330 NEXT:NEXT:LOCATE2,11:PRINT"CARACTER";CN:LOCATE7,13:IF CN
>32 THEN PRINT CHR$(CN) ELSE PRINT CHR$(1)+CHR$(CN+64)
340 RETURN
350 IF Y1=1 THEN 220 ELSE Y1=Y1-1:GOTO 390
360 IF X1=8 THEN 220 ELSE X1=X1+1:GOTO 390
370 IF Y1=8 THEN 220 ELSE Y1=Y1+1:GOTO 390
380 IF X1=1 THEN 220 ELSE X1=X1-1:GOTO 390
390 PUT SPRITE 0,(136+X1*8,39+Y1*8),8,0:FORX=1TO80:NEXT:GOTO
 220
400 LOCATE 15+X1,5+Y1:IF MID$(A$(Y1-1),X1,1)="1" THEN MID$(A
$(Y1-1),X1,1)="0":PRINTCHR$(46) ELSE MID$(A$(Y1-1),X1,1)="1"
:PRINTCHR$(219)
410 NN=CN*8+Y1-1:IF MID$(A$(Y1-1),X1,1)="1" THEN VPOKENN,VPE
EK(NN)OR(2^(8-X1)) ELSE VPOKENN,VPEEK(NN)AND(255-(2^(8-X1)))
420 RETURN
430 LOCATE0,20:PRINT "CHR$ A COPIAR, PULSA RETURN
    ":GOSUB 470:C1=XX
440 LOCATE0,20:PRINT "CHR$ A REEMPLAZAR               ":GOSUB 47
0:C2=XX
450 FORX=0TO7:VPOKEC2*8+X,VPEEK(C1*8+X):NEXT
460 LOCATE0,20:PRINTSTRING$(60,32):RETURN
470 C1$=""
480 X$=INKEY$:IF X$="" THEN 480
490 IF ASC(X$)<32 AND ASC(X$)>27 THEN 480
500 IF ASC(X$)<>13 THEN C1$=C1$+X$:GOTO 480ELSE XX=VAL(C1$)
510 IF XX>255 OR XX<0 THEN 470
520 LOCATE0,20:PRINTSTRING$(60,32):RETURN
```

SPRITES: FIGURAS MOVILES

Las sentencias en BASIC y las variables asociadas con el manejo de sprites, son:

1. SPRITE$(número de patrón)=XYZ$
2. PUT SPRITEn,(X,Y),color,número de patrón
3. PUT SPRITEn,STEP(x,y),color,número de patrón
4. SCREEN Modo,Tamaño de Sprite
5. SPRITE ON/OFF/STOP
6. ON SPRITE GOSUB

Los sprites pueden tener un tamaño de 8x8 ó 16x16 puntos. Además, cada tamaño puede ampliarse al doble. La sentencia SCREEN fija el tamaño a utilizar:

0 = 8*8
1 = 8*8 Ampliado
2 = 16*16
3 = 16*16 Ampliado

Por ejemplo SCREEN 2,2 cambia al modo HRG con sprites de 16*16 puntos.

Un sprite de 8*8 puntos se define de la misma manera que un carácter, utilizando ocho bytes. El patrón de un sprite de 16*16 se crea utilizando cuatro bloques de ocho bytes cada uno:

1 | 8	17 | 24
25 | 32	9 | 16

Para definir un sprite se emplea la sentecia SPRITE$. Cada patrón de sprite está formado por una cadena de 8 ó 32 caracteres, donde cada número de carácter representa un byte del patrón.

Por ejemplo, para definir un sprite de 8*8 equivalente al signo (-) emplearemos:

A$=CHR$(0): B$=CHR$(126)

SPRITE$(0)=A$+A$+A$+B$+B$+A$+A$+A$

El valor 126 se ha obtenido como el equivalente decimal de los bits a nivel 1:

128	64	32	16	8	4	2	1
0	1	1	1	1	1	1	0

Una alternativa al empleo de la sentencia SPRITE$ es la utilización de VPOKE en los primeros ocho bytes de la tabla de patrones de sprites:

FOR X=14336 TO 14343: VPOKE X,0: NEXT: VPOKE 14339,126: VPOKE 14340,126

Si se trabaja con sprites de 8*8 pueden definirse 256 patrones de sprite, en caso contrario, con sprites de 16*16, sólo se admiten 256/4=64 patrones.

Esta capacidad de definir sprites tan pequeños, facilita el uso de juegos de caracteres definidos en sprite. Usados en combinación con la opción de ampliación, puede generarse fácilmente, texto en doble tamaño. Aunque hay una limitación de cuatro letras por línea de pantalla, este procedimiento evita los problemas asociados con la justificación a la izquierda del juego de caracteres residente.

Un sprite puede situarse, o moverse en pantalla, utilizando una de las dos formas de la instrucción PUT SPRITE. El origen de coordenadas se encuentra en la esquina superior izquierda de la pantalla -coordenadas (0,0)-. La esquina inferior izquierda tiene de coordenadas (0,191) y la superior derecha (255,0).

Las coordenadas de PUT SPRITE especificarán el punto situado en la esquina superior izquierda del sprite. Con el objeto de que el sprite pueda entrar, lentamente, en el campo visual, tanto las coordenadas verticales como horizontales, empiezan desde -32. Así, los sprites pueden situarse fuera del campo visualizado de la pantalla.

Las coordenadas que se especifiquen fuera de estos parámetros, y comprendidas entre -32768 y 32767, no producirán error. El sprite será situado en las coordenadas obtenidas como resto entero de la división entera por 256, de las coordenadas anteriores (resultado de aplicar la operación MOD 256).

La única excepción a este procedimiento se produce al señalar como coordenada Y, el valor 208. Cuando el VDP encuentra este valor, el sprite al que se aplica, junto con los sprites de menor prioridad que éste, se borran de la pantalla. Se dispone, así, de un método eficiente para mostrar sprites intermitentes. La forma usual de la sentencia PUT SPRITE es:

PUT SPRITE plano, (X,Y), color, número de patrón

El plano cero es el de mayor prioridad, y, en consecuencia, cualquier parte de un sprite -que no sea transparente- cubrirá cualquier parte de la pantalla (incluso otros sprites) en donde se sitúe. El plano de menor prioridad es el 31. Solamente se permite un sprite por plano.

Se dispone de la misma paleta de colores que con la instrucción COLOR. Unicamente puede definirse un color por sprite. Si no se especifica color, aparecerá el color de primer plano existente. Si se omiten las coordenadas del sprite, éste se situará en la posición (209,0). Como excepción, hay que señalar que si se ha mostrado con anterioridad un sprite en ese plano, el siguiente aparecerá en la última posición del anterior.

Por ejemplo: PUT SPRITE0, (40,40), 10, 0: PUT SPRITE0,, 10, 2 cambiará el patrón de sprite cero por el patrón de sprite dos en el plano cero.

La forma alternativa de la sentencia PUT SPRITE es:

PUT SPRITE plano, STEP(x,y), color, número de patrón

Esta otra forma permite renovar las coordenadas de sprite en relación con su posición anterior.

Los sprites tienen prioridad sobre el plano de caracteres. Cualquier parte -no transparente- de un sprite, ocultará la zona sobre la que se sitúe. Esta circunstancia, al igual que la limitación de un color por sprite, es intrínseca del VDP.

La sentencia PUT SPRITE renueva cuatro bytes de la tabla de atributos de sprite en la VRAM. En aquellos modos que permiten la utilización de sprites, esta tabla empieza en la posición 6912, y tiene 32*4 = 128 bytes de longitud. Los cuatro bytes se asignan con los siguientes valores:

Byte 1: Coordenada Y
2: Coordenada X
3: Número de patrón
4: Bit previo al reloj/color

En BASIC, la coordenada Y puede estar comprendida entre -32 y 209. Sin embargo, como un byte sin signo solamente puede contener valores entre 0 y 255, la tabla utiliza la notación de complemento de dos para las coordenadas negativas (puede verse una explicación de dicha notación en la sección dedicada a Lenguaje Ensamblador Z-80).

Este no es el caso de la coordenada X, gracias al "bit previo al reloj". Si este bit está en nivel 1, la posición del sprite se desplaza 32 puntos hacia la izquierda.

Por ejemplo, para situar un sprite azul en el plano cero, con patrón 1, en las coordenadas (100,100), se puede utilizar la siguiente alternativa a la instrucción PUT SPRITE:

VPOKE 6912, 100: VPOKE 6913, 100

VPOKE 6914, 1: VPOKE 6915, 4

Para mover el sprite 32 posiciones a la izquierda, usaremos:

VPOKE 6913, 68 ó VPOKE 6915, 132

Todos los sprites pueden eliminarse de la pantalla haciendo:

VPOKE 6912, 208

El siguiente ejemplo hace una copia del juego de caracteres en mayúsculas en el área de los patrones de sprite para crear textos especiales:

```
10 SCREEN 1,1: REM Sprites ampliados de 8*8
20 FOR X=0 TO 207
30 VPOKE 14856+X, VPEEK (520+X)
40 NEXT
```

Se han utilizado los números de patrones a partir del 65, para permitir una correspondencia directa entre el código de carácter y el número de patrón. De esta forma, pueden imprimirse fácilmente:

```
10 A$ = "MSX": FOR X=1 TO 3
20 PUT SPRITE X, (18*X,40), 11, ASC(MID$ (A$,X,1) )
30 NEXT
```

Las instrucciones para el control de sprites que quedan por explicar, son las siguientes:

1. ON SPRITE GOSUB
2. SPRITE ON

Estas instrucciones permiten la detección de choques entre sprites y su utilización. Se produce un choque entre sprites cuando se superponen dos zonas no transparentes de los mismos.

Hay dos limitaciones, en la información disponible, sobre la colisión:

1. Solamente se detectan colisiones entre sprites, y no entre sprites y fondo.

2. Los planos de los sprites afectados no pueden obtenerse, ni mediante una función de BASIC, ni a partir de los registros del VDP.

La sentencia SPRITE ON activa el procedimiento de comprobación, que ocurre, tras la ejecución de cada una de las sentencias de BASIC. No se producen comprobaciones en el modo directo, cuando hay un choque, se produce un GOSUB hacia la rutina especificada por la sentencia ON SPRITE GOSUB.

La sentencia SPRITE OFF cancela la comprobación de choques. La sentencia SPRITE STOP suspende la llamada a la subrutina, pero manteniendo la comprobación de colisión. Si se produce un choque, éste se recuerda hasta que se ejecuta una sentencia SPRITE ON, momento en el cual se realiza la llamada a la subrutina.

Al entrar en la subrutina, se ejecuta automáticamente una sentencia SPRITE STOP. Esta es posteriormente anulada por una sentencia SPRITE ON al encontrar la instrucción RETURN, a menos que la propia rutina contenga una sentencia SPRITE OFF. Hay que señalar que la instrucción CLEAR provoca una sentencia SPRITE OFF.

Frecuentemente, la naturaleza del movimiento de los sprites sigue unas pautas repetitivas. Esto supone que sus posiciones sean incrementadas o disminuidas en intervalos fijos. El BASIC MSX incorpora un potente juego de instrucciones que desvían la ejecución del programa a determinadas subrutinas en intervalos regulares. El período de tiempo de estos intervalos se fija en unidades de 1/50 seg. (la frecuencia de interrupción del sistema del VDP).

El formato de estas instrucciones es similar al de las sentencias de tratamiento de colisión entre sprites:

1. INTERVAL ON
2. ON INTERVAL=X GOSUB

De nuevo, están las variantes INTERVAL OFF e INTERVAL STOP, la última de las cuales se ejecuta al entrar en la rutina de control.

Es posible que se detecte una pequeña variación en el período, especialmente en intervalos cortos, debido a la comprobación que se produce después de cada instrucción BASIC.

Dos juegos de instrucciones similares a los anteriores, son:

1. ON STOP GOSUB junto con STOP ON/OFF/STOP
2. ON KEY GOSUB linea x, linea y, ... con KEY(x) ON/OFF/STOP

La estructura ON STOP GOSUB / STOP ON debe usarse con precaución, porque debe impedir el retorno desde el modo de programa al modo directo. Por ejemplo:

```
10 ON STOP GOSUB 40: STOP ON
20 PRINT "SIN PARAR": GOTO 20
40 RETURN
```


Este programa no puede detenerse. Si se pretende que la ejecución de un programa no pueda interrumpirse, las sentencias de la linea 10 deben de aparecer después de cada sentencia CLEAR.

PROGRAMA EJEMPLO: DISEÑO DE SPRITES

A continuación se encuentra el listado de un programa muy útil, que permite diseñar sprites.

```
10 '*******************************
20 '****** DISEÑO DE SPRITES *****
30 '*******************************
40 '
50 'UTILIZA LAS TECLAS DE CURSOR PARA     MOVERTE POR LA MATR
IZ.
60 '
70 'PARA MARCAR EL PUNTO A LA          IZQUIERDA DEL CURSO
R PULSA LA        BARRA DE ESPACIO.                   PARA
TERMINAR Y OBTENER LOS            VALORES FINALES PULSA CTRL/S
TOP.
80 '
90 KEYOFF:FORX=0TO32:VPOKEX+14336,0:NEXT
100 ON STOP GOSUB300:STOP ON
110 SCREEN1,2:COLOR14,1,1:CLS
120 A$=STRING$(16,CHR$(250)):FORX=4TO19:LOCATE11,X:PRINTA$:N
EXT
130 X=12:Y=4:LOCATEX,Y,1:PUT SPRITE0,(20,100),14,0
140 Q=0:ON STRIG GOSUB 250:STRIG(0) ON
150 FORL=1 TO 60:NEXT:D=STICK(0):ON D GOTO170,150,190,150,21
0,150,230
160 GOTO 150
170 IF Y<>4 THENY=Y-1:LOCATEX,Y,1 ELSE 150
180 IF Y<>11 THEN 150 ELSE Q=Q-1:GOTO 150
190 IF X=27 THEN 150 ELSE X=X+1:LOCATEX,Y,1
200 IF X<>20 THEN 150 ELSE Q=Q+2:GOTO 150
210 IF Y=19 THEN 150 ELSE Y=Y+1:LOCATEX,Y,1
220 IF Y<>12 THEN 150 ELSE Q=Q+1:GOTO 150
230 IF X=12 THEN 150 ELSE X=X-1:LOCATEX,Y,1
240 IF X<>19 THEN 150 ELSE Q=Q-2:GOTO 150
250 IF VPEEK(6145+32*Y+X)=250 THEN VPOKE 6145+32*Y+X,219:AA=
2 ELSE VPOKE 6145+32*Y+X,250:AA=4
260 BY=14336+Q*8:IF Q=0 OR Q=2 THEN BY=BY+Y-4 ELSE BY=BY+Y-1
2
270 IF Q=0 OR Q=1 THEN BI=19-X ELSE BI=27-X
280 VL=2^BI:IF AA=2 THEN VPOKEBY,(VPEEK(BY) OR VL) ELSE VPOK
EBY,(VPEEK(BY)AND(255-VL))
290 RETURN
300 CLS
310 FOR X=0TO15:LOCATE2,X+2:PRINTVPEEK(14336+X):LOCATE8,X+2:
PRINTHEX$(VPEEK(14336+X)):LOCATE12,X+2:PRINT VPEEK(14352+X):
LOCATE18,X+2:PRINT HEX$(VPEEK(14352+X)):NEXT
```

GRAFICOS DE ALTA RESOLUCION

El BASIC MSX tiene siete instrucciones que sólo pueden utilizarse en los dos modos gráficos:

1. CIRCLE 2. DRAW 3. LINE 4. PAINT

5. PSET 6. PRESET 7. POINT

Sin embargo, estas instrucciones producen resultados poco precisos en el modo multicolor, que tiene una resolución máxima de bloques de 4*4 puntos.

No es posible imprimir caracteres en la pantalla de gráficos en la forma habitual. Por el contrario, los caracteres deben ser enviados a la pantalla como datos de archivo:

```
10 OPEN "GRP:" FOR OUTPUT AS #1
20 PRESET (40,40)
30 PRINT #1, "Texto en modo gráfico"
```

Hay que hacer notar que a menos que se haya declarado más de un archivo con la sentencia MAXFILES, el número de archivo debe ser el 1. En el modo multicolor, cada letra tendrá, aproximadamente, 2.5 cm. de lado.

La instrucción más utilizada para gráficos es DRAW, que tiene el formato:

DRAW "instrucciones de microlenguaje de gráficos"

Las trece instrucciones disponibles constituyen lo que Microsoft ha bautizado como "Microlenguaje de gráficos". Estas instrucciones permiten trazar líneas en determinados intervalos, tanto hacia arriba, hacia abajo, a la izquierda, a la derecha o en diagonal, desde el anterior punto de referencia. Pueden utilizarse cualquiera de los 16 colores. La sentencia se explica mejor mediante un ejemplo:

DRAW "D40R40U40L40"

Esta sentencia dibuja un cuadrado con un trazo hacia abajo de 40 unidades (D40), después un trazo hacia la derecha de 40 unidades (R40), otro hacia arriba de 40 unidades (U40) y, por último, hacia la izquierda también de 40 unidades (L40). Las instrucciones de las distintas direcciones son:

```
            U
     H             E
L           +             R
     G             F
            D
```

El color a utilizar se fija mediante Cx.

Para dibujar una línea hasta una posición especificada mediante coordenadas absolutas, se emplea M X,Y Alternativamente, para dibujar hasta una posición dada mediante coordenadas relativas a la posición anterior, hay que colocar delante de los valores de X e Y, tanto el signo "+" como el "-" según corresponda.

Las unidades de movimiento se determinan mediante la instrucción Sn, donde "n" debe estar comprendida entre 0 y 255 (por omisión, se toma el valor 4). El valor de "n" se divide por 4 para obtener el número de puntos de cada unidad.

Si una instrucción especifica una posición fuera de la pantalla, pero entre -32768 y 32767, no se señala error, adoptándose la posición más extrema, dentro de las posibles.

Hay dos variantes opcionales de cada instrucción de movimiento:

1. Empleando el prefijo "B", se consigue el desplazamiento del cursor gráfico, a la distancia especificada, sin trazar línea alguna.

2. El prefijo "N" produce el desplazamiento del cursor gráfico a la posición inicial, al completar la subinstrucción.

Los ejes directores pueden girar en sentido contrario a las agujas del reloj, en múltiplos de 90°, para ello se utiliza la instrucción An. El valor numérico "n" debe ser entero, y estar comprendido entre 0 y 3, según las direcciones:

```
    0
1   +   3
    2
```

Por lo tanto, DRAW "A3L40" dibujará una línea hacia la derecha de 40 unidades.

La instrucción X permite incluir una variable de cadena dentro de la de la instrucción.

Se emplea como prefijo de la variable, que debe ir seguida por un punto y coma (;). Por ejemplo:

ST$="M-20,+40": DRAW "XST$;"

Este comando no es necesario si no se utilizan delimitadores:

DRAW ST$

También puede utilizarse una variable numérica en una cadena de instrucción. La variable debe ir precedida del signo igual (=) y seguida por un punto y coma (;):

DRAW "U=Y;"

El formato de la instrucción LINE es:

LINE (X1,Y1) - (X2,Y2), color

Esta instrucción dibuja una línea entre las dos coordenadas. Cada coordenada puede, a su vez, especificarse mediante la variante STEP de coordenadas relativas.

Si se añade una ",B" al final de la instrucción, se obtiene un rectángulo. Si se pone ",BF" se dibuja un rectángulo, que se rellena de color.

La instrucción CIRCLE permite realizar con facilidad diagramas complejos. Se puede dibujar cualquier tipo de elipse, tanto enteras como arcos de elipse. En este último caso, los extremos del arco pueden cerrarse en abanico. El formato básico de esta instrucción es:

CIRCLE (X,Y), radio

con las opciones:

, color, ángulo de comienzo, ángulo final, aspecto.

Las coordenadas X,Y especifican el centro del círculo. La forma STEP (X,Y) permite definir dicho punto en relación con la posición anterior. Los ángulos de comienzo y final admiten argumentos comprendidos entre 0 y 2*PI. También se aceptará un valor negativo, pero se dibujarán líneas adicionales entre el centro de la elipse y los puntos de comienzo y final. El aspecto es una relación entre el radio vertical y el horizontal.

La instrucción PAINT adopta la forma:

PAINT (X,Y) ó PAINT (X,Y), color

Y rellenará de color cualquier figura cerrada a la que pertenezca el punto definido por (X,Y). Si no se especifica color, se empleará el color de primer plano. En el modo HRG (gráficos de alta resolución) el color, tanto de la línea de cierre de la figura, como de la "pintura", debe ser el mismo, en caso contrario, la totalidad de la pantalla se rellenará de color. En el modo multicolor, se puede utilizar el formato siguiente:

PAINT (X,Y), color de relleno, color de borde.

Para producir un círculo relleno de rojo en el modo HRG, utilizaremos:

SCREEN 2: COLOR 8,1,1: CIRCLE (80,80),40: PAINT (80,80)

Las dos instrucciones restantes son:

PSET (X,Y),color

PRESET (X,Y), color

que son idénticas, en el sentido de que ambas fijan el color del punto especificado. El color puede omitirse, en cuyo caso PSET toma el color de primer plano y PRESET el color de fondo.

EJEMPLO DE PROGRAMA: TABLERO DE APUNTES

El siguiente programa nos enseña el uso de gráficos de alta resolución.

```
10 '***********************
20 '** TABLERO DE APUNTES **
30 '***********************
40 '
50 'MANDO DE JUEGOS EN EL        PUERTO 1. DIBUJA PRE-
  SIONANDO EL BOTON DE         DISPARO
60 '
70 'F1 PARA CAMBIAR COLOR;       SEGUIDO POR EL NO. DE
  COLOR -2 CARACTERES-
80 'F2 PARADIBUJAR UN CIRCULO    SEGUIDO POR EL RADIO
  -2 CARACTERES-
90 'F3 PARA RELLENAR UNA         FIGURA; CON BORDE DEL
  MISMO COLOR
100 'F4 PARA GUARDAR EL           DIBUJO; 2 MINUTOS PARA
   GRABAR
110 'F5 PARA CARGAR UNA PAN-      TALLA DE LA CASSETTE;
   PRESIONAR PLAY DESPUES       DE F5
 2 MINUTOS PARA CARGAR
120 '
130 FORX=38000!TO38024!:READ A$:A=VAL("&H"+A$):POKEX,A:NEXT:
DEFUSR=38000!
140 DATA 21,00,00,11,40,9C,01,00,18,CD,59,00,21,00,20,11,40,
B4,01,00,18,CD,59,00,C9
150 FORX=38200!TO38224!:READ A$:A=VAL("&H"+A$):POKEX,A:NEXT:
DEFUSR2=38200!
160 DATA 21,40,9C,11,00,00,01,00,18,CD,5C,00,21,40,B4,11,00,
20,01,00,18,CD,5C,00,C9
170 ON KEY GOSUB 350,360,370,380,410:KEY(1) ON:KEY(2) ON:KEY
(3) ON:KEY(4) ON:KEY(5) ON
180 ON STRIG GOSUB 180,240:STRIG(1) ON
190 COLOR 10,1,10:SCREEN 2,0:CLS:X=125:Y=95
200 FOR Z=14336 TO 14343:READ ZZ:VPOKEZ,ZZ:NEXT:C=10
210 DATA 0,32,32,248,32,32,0,0,0,0
220 PUT SPRITE0,(X-2,Y-4),10,0:IF A=1 THEN RETURN ELSE A=1
230 GOTO 230
240 DN=STICK(1):IF DN=0 THEN 240
250 ON DN GOTO 260,270,280,290,300,310,320,330
260 IF Y=0 THEN 240 ELSE Y=Y-1:GOTO 340
270 IF X=255 OR Y=0 THEN 240 ELSE X=X+1:Y=Y-1:GOTO 340
280 IF X=255 THEN 240 ELSE X=X+1:GOTO 340
290 IF X=255 OR Y=191 THEN 240 ELSE X=X+1:Y=Y+1:GOTO 340
300 IF Y=191 THEN 240 ELSE Y=Y+1:GOTO 340
310 IF X=0 OR Y=191 THEN 240 ELSE X=X-1:Y=Y+1:GOTO 340
320 IF X=0 THEN 240 ELSE X=X-1:GOTO 340
330 IF Y=0 OR X=0 THEN 240 ELSE Y=Y-1:X=X-1
340 IF C<>0 THEN PSET (X,Y),C:GOTO 220 ELSE 220
350 PUT SPRITE 2,(20,20),10,0:PUT SPRITE 4,(28,20),10,0:A$=I
NPUT$(1):PUT SPRITE 2,(0,0),,0:B$=INPUT$(1):PUT SPRITE 4,(0,
0),,0:C=VAL(A$+B$):RETURN
360 PUT SPRITE 2,(20,80),10,0:PUT SPRITE 4,(28,80),10,0:A$=I
NPUT$(1):PUT SPRITE 2,(0,0),,0:B$=INPUT$(1):PUT SPRITE 4,(0,
0),,0:R=VAL(A$+B$):CIRCLE(X,Y),R,C,,,1.4:RETURN
```

```
370 PAINT (X,Y),C,C:RETURN
380 Z=USR(2)
390 SCREEN 1:CLS:LOCATE 2,2:PRINT"PREPARA LA CINTA PARA GRAB
ACION Y PULSA RETURN"
400 A$=INPUT$(1):IF ASC(A$)<>13 THEN 400 ELSE BSAVE "CAS:",4
0000!,52288!:STOP
410 PUT SPRITE 2,(20,20),10,0:BLOAD"CAS:":Z=USR 2(2):PUT SPR
ITE 2,(0,0),,0:RETURN
```

SONIDO

Para generar un "bip" puede utilizarse la sentencia BEEP ó PRINT CHR$(7) Los sonidos más complejos se generan, usando la sentencia PLAY con un macrolenguaje, o modificando directamente los registros del PSG (generador programable de sonido) con la instrucción SOUND. Esta última se utiliza de la forma siguiente:

SOUND registro #, valor

Para más detalles sobre las funciones de los registros, véase el capítulo 7.

La sentencia PLAY puede ir seguida por un máximo de tres cadenas, cada una de las cuales es una secuencia de instrucciones de una letra dirigidas a cada uno de los canales de sonido respectivos (el formato es similar al de la instrucción DRAW).

Por ejemplo, para tocar una única nota musical por el canal 2, debe hacerse:

PLAY "", "N20"

Debido a que los valores por omisión se fijan automáticamente para la mayoría de los parámetros, no es necesario utilizar largas secuencias de inicialización.

Hay dos métodos para especificar la nota a tocar:

1. Nx donde x está comprendida entre 0 y 96. Si se utiliza el 0, se produce un silencio.

2. Fijando la octava mediante Ox, donde x debe estar comprendida entre 1 y 8, siendo 4 el valor tomado por omisión y la nota mediante las iniciales A-G. Por ejemplo:

PLAY "O6ABCD" tocará una secuencia de cuatro notas en la sexta octava.

Si esta instrucción fuera seguida por PLAY "BCD" estas notas también se tocarían en la sexta octava.

Los bemoles y sostenidos (solamente aquellos disponibles en piano) se producen utilizando los signos # ó + después de la nota para los sostenidos, y el signo - para los bemoles.

El volumen por omisión toma el valor 8 en una escala desde 0 a 15, y se cambia con la instrucción Vx.

La longitud de cada nota se determina con Lx, donde x debe estar comprendida entre 1 y 64. El valor 1 producirá una redonda, un valor de 4 generará una negra, etc. Para obtener una nota más larga que 1, debe cambiarse el parámetro tempo, utilizando la instrucción Tx (valor por omisión 120). El valor x puede estar comprendido entre 32 y 255, y determina el número de negras tocadas en un minuto. Por ejemplo, para hacer que todas las notas tengan una duración doble de la habitual, se hará T60.

Si solamente es necesario cambiar la longitud de una única nota, el valor inverso de la duración deseada, debe situarse a continuación de la nota. Por ejemplo B#8 ó D32.

Se puede utilizar una variable con todas las instrucciones de una única letra, para ello, se emplea el formato:

PLAY "N=X;L=Y;N=X;"

Para simplificar la transcripción, Microsoft incorporó la capacidad de alargar una nota mediante un punto (.) con lo que su duración pasa a ser de vez y media la que tenía. Una alternativa a la utilización de NO para producir una pausa, es la instrucción Rx (x=4 por omisión) . Se produce así un descanso de la longitud especificada por x, que debe encontrarse entre 1 y 64, y que es interpretada en forma semejante al parámetro de la instrucción L.

Las dos instrucciones de una sola letra restantes, Mx y Sx, permiten generar efectos de sonido especiales alterando la envolvente del sonido producido.

De esta forma, el volumen puede variar de una manera preestablecida a lo largo de la duración de la nota. La forma de la envolvente comienza con una de estas dos secuencias:

1. El volumen aumenta desde 0 al máximo. La envolvente tiene un valor de ataque de 4.

2. El volumen disminuye desde un máximo a 0. La envolvente tiene un valor de ataque de 0.

El resto de la secuencia de envolvente se determina mediante otros tres parámetros. El valor requerido para cada uno se suma al del valor de ataque inicial. El valor obtenido se usa con la instruccion S para generar la forma de sonido requerida.

1. Continuado: si el sonido va a terminar después del primer ciclo, se pone a cero, en caso contrario, se toma el valor 8.

2. Mantenido: si se utiliza el valor 1, la secuencia de ataque se repite continuamente en cada ciclo. Si se emplea el valor cero, el volumen se mantiene en el nivel del final del primer ciclo.

3. Alternado: un valor de 2 altera el nivel del volumen al final de cada ciclo. No se producen cambios si se utiliza el valor 0.

Cada uno de los anteriores parámetros, tiene dos valores posibles, que permiten un total de 16 combinaciones de ciclo y forma de envolvente. Sin embargo, existe una duplicidad de valores, y solamente se pueden conseguir ocho patrones diferentes.

Estos patrones, junto con los valores de los parámetros que los determinan, pueden encontrarse en la página quinta del apéndice F. Los valores asumidos por omisión son:

Forma: 1
Modulación: 255

Por ejemplo, para producir notas que aumentan a un volumen máximo y se mantienen a lo largo de su duración, se utilizan los siguientes valores de parámetros:

1. Ataque: 4
2. Continuado: 8
3. Mantenido: 1
4. Alternado: 0

Total = 13: PLAY "S13 ...

La duración del primero y de los siguientes ciclos, se fija con la instrucción M, que admite valores enteros entre 1 y 65535 inclusive.

Un ultimo aspecto de la instrucción PLAY (al igual que la instrucción DRAW), pueden ejecutarse cadenas predefinidas mediante la instrucción X:

A$ = "N40N20N40": PLAY "XA$;"

Siempre que se utilice una variable, debe ir seguida por un punto y coma (;) La situación (si se está ejecutando música o ha finalizado la ejecución), de uno o de todos los canales puede obtenerse con la función PLAY:

PLAY (x)

El valor de x debe estar entre 0 y 3. Si se comprueba el canal 1, 2 ó 3, y si el canal está activo, se obtiene el valor -1; en caso contrario, se obtiene el valor 0. Si se ejecuta PLAY (0), los estados de los tres canales se operan entre sí, mediante el operador lógico OR, y el resultado es devuelto por dicha función.

ALMACENAMIENTO DE PROGRAMA

En todos los ordenadores MSX con al menos 32 K de RAM, el almacenamiento de programas en BASIC comienza a partir de la posición 32768 (&H8000). Cada linea de programa se almacena en una forma condensada (diferente de la que fue escrita). Las palabras clave se sustituyen por signos, mientras que los nombres de variables y los símbolos, se almacenan directamente. Cada línea de programa es precedida por dos pares de bytes. El segundo par contiene el número de línea, y el primero, la dirección de la línea siguiente. Las líneas se separan por un byte nulo, y el final de programa se indica mediante dos bytes adicionales del mismo valor.

En posiciones de memoria superiores a las de programa, encontramos una tabla de variables.

En esta tabla se almacenan aquellas variables, que no son de conjunto, utilizadas en el programa. Por encima de esta tabla de variables sencillas, se encuentra otra de variables de conjunto.

Por ejemplo, si escribimos el siguiente programa:

```
10 FOR X=32768! TO 40000!
20 PRINTX,PEEK(X)
40 NEXT
```

El almacenamiento queda organizado como sigue:

POSICION	VALOR	COMENTARIO
32768	0	
32769	23	dirección de la próxima
32770	128	línea
32771	10	numero de línea
32772	0	
32773	130	signo de FOR
32774	32	código de espacio
32775	88	X
32776	239	signo de =
32777	29	constante de precisión sencilla
32778	69	byte de exponente precisión sencilla
32779	50	
32780	118	mantisa de precisión sencilla
32781	128	
32782	32	código de espacio
32783	217	signo de TO
32784	32	código de espacio
32785	29	constante de precisión sencilla
32786	69	byte de exponente de precisión sencilla
32787	64	
32788	0	mantisa de precisión sencilla
32789	0	
32790	0	marca de fin de línea
32791	36	dirección de la próxima
32792	128	línea
32793	20	número de línea
32794	0	
32795	145	signo de PRINT
32796	88	X
32797	44	coma
32798	255	signo de PEEK
32799	151	
32800	40	(
32801	88	X
32802	41	)
32803	0	marca de fin de línea
32804	42	direccion de la próxima
32805	128	línea
32806	40	numero de línea
32807	0	
32808	131	signo de NEXT
32809	0	marca de fin de línea
32810	0	marca de fin de
32811	0	programa

Las constantes se almacenan en el formato requerido. En el ejemplo los parámetros del bucle FOR-NEXT, se almacenan en precisión sencilla.

La tabla de variables simples se mueve siempre que se añadan o se eliminen líneas de programa Las variables se almacenan en el orden en que aparecen en el programa. El primer byte de cada una indica el tipo de variable:

8: Precisión doble
4: Precisión sencilla
3: Cadena
2: Entera

Los dos bytes siguientes son los códigos de los dos primeros caracteres del nombre de la variable. Los restantes, difieren según el tipo de variable:

1. Entera: el valor existente se almacena en forma binaria inversa, de dos bytes con signo.

2. Cadena: se utilizan tres bytes para almacenar los datos de la cadena. El primero indica el número de caracteres. Los siguientes contienen la dirección donde se encuentra la cadena. La dirección tiene el byte de orden mas bajo y de orden más alto invertidos.

3 Precision sencilla: el valor se representa mediante un byte para el exponente, y mediante una mantisa de seis dígitos, almacenada en decimal codificado en binario de tres bytes. El bit más significativo de la mantisa señala el signo del valor: 1=negativo.

4. Precisión doble: similar al de variable de precisión sencilla, con la diferencia de utilizar una mantisa de siete bytes.

El byte de exponente, cuya posición se obtiene mediante la función VARPTR, tiene sumado el valor &H40.

Cada elemento de la tabla de variables de conjunto que se encuentra a continuación, comienza asímismo, con tres bytes, que contienen el tipo de variable y su nombre. Sin embargo, delante de los datos se encuentra un encabezamiento, que consta de tres secciones:

1. Una parte inicial de dos bytes que contiene el número de bytes restantes del conjunto.

2. Un único byte que almacena el número de dimensiones.

3. Una secuencia de valores de dos bytes que indica el tamaño de cada dimension.

CAPITULO 3

VOCABULARIO BASIC MSX

ABS(Y)

Esta funcion devuelve el valor absoluto de la expresión numérica Y

Ejemplo: ABS(-4) resulta 4.

ASC(Y$)

Entrega el código de carácter del primer carácter de la cadena Y$. Un símbolo gráfico genera el valor 1. Una cadena nula produce un mensaje de error. La lista completa de códigos de carácter, puede encontrarse en el apéndice A.

Ejemplo: ASC("ABC") resulta 65.

ATN(Y)

Esta funcion calcula el arcotangente de la expresión Y. El resultado se obtiene en radianes, entre -PI/2 y PI/2. Si Y es una variable, puede ser de cualquier precision, sin embargo, el cálculo se realiza en doble precision.

Ejemplo: C=ATN(40) da como resultado 0.67474094222354

AUTO

Variantes: AUTO
AUTO número de linea
AUTO ,incremento
AUTO número de línea,incremento

Valores por omisiones: número de linea ... 0
incremento 10
incremento y
número de linea ... 10

El siguiente número de línea se imprime después de cada retorno de carro.

Las opciones permiten especificar el número de línea inicial y el incremento. La secuencia finaliza pulsando CTRL C ó CTRL STOP. Se imprime un asterisco detrás de cualquier número de línea existente. Pulsando la tecla RETURN exclusivamente, la línea en edición quedará inalterada.

Ejemplo: AUTO 200,20 generará la secuencia 200, 220, 240 ...

BASE(Y)

Es una variable especial, que nos permite obtener las posiciones de la RAM de video en que se encuentran las tablas utilizadas por el VDP para generar la pantalla. Los valores de Y comprendidos entre 0 y 19, producen la posición de comienzo de una tabla en un modo particular. Veáse el apéndice C para más detalles. Cada modo de pantalla tiene una tabla de nombres y de patrones generadores, que contienen los datos del plano de fondo.

Con la excepción del modo de texto de 40 columnas, los modos de pantalla tienen, además, una tabla de patrones de sprite, y una tabla de atributos de sprite. La posición de cada una de las tablas de sprite no cambia entre los modos:

Tabla de atributos de sprite: 6912
Tabla de patrones de sprite: 14336

Ejemplo: BASE(0) entrega 0, la base de la tabla de nombres en el modo de texto.

BEEP

Genera un sonido ("bip").

BIN$(Y)

Esta función produce una cadena conteniendo el equivalente binario de la expresion decimal Y.

Y debe estar comprendida entre -32768 y 65535. Un valor negativo se expresa en la notación de complemento de 2.

Ejemplo: BIN$(-1) entrega "1111111111111111"

BLOAD

Variantes: BLOAD "CAS:"
BLOAD "CAS:nombre de archivo"
BLOAD "CAS:",R
BLOAD "CAS:",desplazamiento

Carga un programa en lenguaje máquina desde el cassette (único dispositivo soportado por la versión 1 de BASIC), en la posición desde la que

fué guardado.

La opción R ejecuta una llamada a la posición especificada en BSAVE al completarse la carga del programa. La opción de desplazamiento, cambia la posición en que se almacena el programa en memoria, con respecto a la posición que tenia cuando fue guardado.

Ejemplo: BLOAD "CAS:",R,&H20 cargará y ejecutará el siguiente programa en lenguaje máquina encontrado en el cassette. El programa se cargará 32 posiciones más arriba de donde se encontraba cuando fue guardado.

BSAVE "CAS:", principio, fin

Variantes: BSAVE "CAS:nombre de archivo", principio, fin
BSAVE "CAS:", principio, fin, posición de llamada

Esta instrucción se utiliza para guardar un bloque de memoria en el cassette. Una de las opciones especifica la posición donde comienza la ejecución del programa, si éste es cargado mediante la instrucción: BLOAD "CAS:",R

Si se utiliza BLOAD "CAS:",R sin que haya sido especificada la posición de llamada en el momento de almacenar el programa, la ejecución del programa comienza en la primera posición ocupada.

Ejemplo: BSAVE "CAS:",34000,38000,36000 guarda el bloque de 4K entre las posiciones 34000 y 38000, inclusive. Si el programa se carga usando la opción BLOAD "CAS:",R en el momento de completarse la carga se efectúa una llamada a la posición 36000.

CALL nombre de sentencia

Variantes: CALL nombre de sentencia (argumento)
CALL nombre de sentencia (argumento, argumento, ...)

La instrucción CALL puede sustituirse por un subrayado (_)

CALL se utiliza para ejecutar una instrucción de extensión provista por un cartucho de ROM.

Ejemplo: CALL SPSET (176,32)

CDBL(Y)

La expresión numérica Y se convierte a un número de doble precisión. Y puede ser de cualquier tipo numérico.

CHR$(Y)

Entrega una cadena formada por un único carácter correspondiente al código Y. Para imprimir un simbolo gráfico, debe hacerse previamente PRINT CHR$(1)

Ejemplo: CHR$(65) entrega A

CINT(Y)

Convierte Y a un número entero, truncándola. Se produce un error si Y no está comprendida entre -32768 y 32767.

Ejemplo: Y=4.8:?CINT(Y) dará 4

CIRCLE(X,Y),radio

Variantes: CIRCLE(X,Y), radio, color
CIRCLE(X,Y), radio, color, ángulo de comienzo, ángulo de fin
CIRCLE(X,Y), radio,, ángulo de comienzo
CIRCLE(X,Y), radio,,, ángulo de fin
CIRCLE(X,Y), radio, color, ángulo de comienzo, angulo de fin, aspecto

El ángulo de comienzo por omisión es 0, y el ángulo de fin es 2PI.

El centro del círculo, especificado en coordenadas absolutas, mediante (X,Y) puede ser, asímismo, especificado en forma relativa, con respecto a la anterior posición, utilizando STEP(X,Y)

Si no se especifica color, se adoptará el color de primer plano. Si se utilizan las opciones de ángulo de comienzo y fin, el parámetro estará en radianes y comprendido entre -2PI y 2PI. El aspecto determina la relacion horizontal/vertical de los radios de la elipse.

Ejemplo: CIRCLE(20,20), 10, 10, 3.142, 2 produce media elipse desde PI a 2PI.

CLEAR

Variantes: CLEAR
CLEAR espacio para cadenas
CLEAR espacio para cadenas, posición superior de memoria

El valor asumido por omisión de la posición superior de memoria es &HF380

Con esta instrucción, todos los archivos abiertos se cierran, las variables numéricas se asignan con el valor 0, y las cadenas, con la cadena nula.

El espacio para cadenas por omisión es de 200 bytes. La posición superior de memoria (Memtop) es la posicion más alta en la memoria, que puede ser utilizada por BASIC. Esta posición puede rebajarse para permitir que un programa en lenguaje máquina sea alojado sobre el area de BASIC. Así, se asegura que el código maquina no sera alterado con datos de BASIC.

Todas las sentencias ON-GOTO/GOSUB también se anulan mediante esta instrucción.

Ejemplo: CLEAR 400,44000 fija el espacio disponible para cadenas en 400 bytes, y altera el "techo" de BASIC a 44000.

CLOAD

Variante: CLOAD "nombre de archivo"

CLOAD cierra todos los archivos de programa y anula todas las variables antes de cargar el siguiente archivo de programa desde el cassette. Si se coloca un nombre de archivo, solamente se consideran significativos los seis primeros caracteres.

CLOAD?

Variante: CLOAD? "nombre de archivo"

Compara el archivo de programa existente en memoria con el siguiente archivo de programa en el cassette. Una diferencia entre ambos produce el mensaje "Verify error".

CLOSE

Variante: CLOSE
CLOSE #Y
CLOSE #Y, #Z

Los símbolos numéricos (#) son opcionales.

Salvo que se especifique un numero de archivo, todos los archivos abiertos son cerrados, y cualquier memoria intermedia asociada se anula. Si la memoria intermedia es de salida, los datos en ella contenidos son enviados.

Ejemplo: CLOSE 4 cerrará y liberará la memoria intermedia asociada al canal 4.

CLS

Borra la pantalla, dejándola del color del fondo especificado en la sentencia COLOR. En modo de texto, el cursor se desplaza a la esquina superior izquierda. CLS opera en todos los modos.

COLOR color de primer plano, color de fondo, color de borde

Variante: COLOR , color de fondo
COLOR , color de fondo, color de borde
COLOR ,, color de borde
COLOR color de primer plano,, color de borde

La instrucción COLOR puede utilizarse en todos los modos de pantalla. En cualquiera de los dos modos de texto, el color de fondo se asume inmediatamente.

Los modos de gráficos de alta resolución y multicolor, precisan de la instrucción CLS para realizar el cambio. En el modo de texto de 40 columnas, el color de borde es el mismo del color de fondo. La tabla de color de la VRAM se modifica en todos los modos.

Ejemplo: COLOR 4,10,14 pondrá el borde de color gris, el plano de fondo en amarillo oscuro y la tinta en azul oscuro.

CONT

Continua la ejecución del programa después de una detención.

COS(Y)

Permite obtener el coseno de un ángulo, dado en radianes, mediante la expresión Y.

CSAVE "nombre de archivo"

Variante: CSAVE "nombre de archivo", velocidad de transmisión

El programa existente en memoria, se graba en cassette. Solamente los seis primeros caracteres del nombre de archivo son significativos. El programa puede grabarse, tanto a una velocidad de 1200 baudios (por omisión), o de 2400.

Ejemplo: CSAVE "nombre", 1 ... 1200 baudios
CSAVE "nombre", 2 ... 2400 baudios

CSNG(Y)

Convierte Y a un número de precisión sencilla.

CSRLIN

Calcula la posición vertical del cursor de texto, entre 0 y 23.

Ejemplo: FILA = CSRLIN

DATA

Es un almacén de datos al que se accede mediante una sentencia READ. Los datos numericos y de cadena pueden mezclarse en una linea. Los datos deben ir separados por comas. Las cadenas no necesitan encerrarse entre comillas, salvo que contengan comas, dos puntos, punto y coma o espacios significativos al comienzo o final.

Ejemplo: DATA 1.4,6,8.0,,,,4,2

DEF FN Y(A) = expresión

Variante: DEF FN Y(A,B,C) = expresión

Una función se define mediante una expresión que contiene los parámetros entre paréntesis. Cuando dicha función sea calculada, los valores dados entre paréntesis, serán sustituidos en la expresión. Estos valores deben ser del mismo tipo que los parámetros usados en la definición. Pueden usarse hasta un máximo de ocho parámetros en la definición.

Ejemplos: DEF FN Y(S) = 2*S:T = 4:W = FN Y(T) da como resultado 8
DEF FN Y(S,A) = 2*S+(A+4):T = 2:R = 4:N = FN Y(T,R) calcula 2*2 + (4+4)

DEFDBL Y DEFINT Y DEFSNG Y DEFSTR Y

Variantes: DEFINT W-Z
DEFDBL A,B-S

Cada una de estas instrucciones declara cualquier variable (tanto simple como de conjunto), que empiece con la letra dada como de ese tipo particular de variable. Esta asignación puede ser evitada para una variable individual, mediante una declaración de tipo, escrita a continuación.

El tipo de variable declarada es, en cada instrucción, el siguiente: DEFDBL doble precisión, DEFINT entera, DEFSNG precisión sencilla, DEFSTR cadena.

Ejemplo: DEFINT A,B define todas las variables que empiezan por A ó B como variables enteras.

DEFUSRx = dirección

DEFSR especifica la direccion de comienzo de una rutina en código máquina. La rutina se ejecuta mediante la función USR. El entero x debe estar comprendido entre 0 y 9, si se omite, se toma el valor 0.

Ejemplo: DEFUSR2 = &H4000

DELETE número de línea-número de linea

Variantes: DELETE número de línea
DELETE - número de línea

Es una instruccion de edición, que permite borrar una parte del programa Cualquier número de línea puede omitirse, pero si se determina, debe existir en el programa.

DIM Y(x)

Variantes: DIM Y(x),Z(n)

Si se utilizan más de once elementos por variable, debe reservarse espacio para ellos, mediante la sentencia DIM. Un conjunto de cadenas asignará 255 bytes por cada elemento.

Ejemplo: DIM Y(20),S$(12) permitirá utilizar los elementos comprendidos entre Y(0) - Y(20) y S$(0) - S$(12).

DRAW "cadena de instrucciones"

Es la principal instrucción gráfica, con un subconjunto de trece instrucciones. Veáse el capítulo 2 para más detalles.

END

Finaliza la ejecución de programa, y cierra todos los archivos abiertos No se imprime mensaje de ruptura.

EOF (número de archivo)

Se utiliza para comprobar si se ha alcanzado el fin de un archivo cuando se reciben datos de un archivo secuencial. Entrega el valor 0 excepto al final del archivo, en cuyo caso se obtiene -1.

Ejemplo: IF EOF(8) = -1 THEN 200

ERASE variable de conjunto

Variante: ERASE variable de conjunto, ...

Elimina los conjuntos que empiezan con las letras especificadas. Para volver a utilizarse deben redimensionarse los conjuntos.

Ejemplo: ERASE X,Y borrará XC(16) e YJ(14)

ERL

Al producirse un error, ERL contendrá el número de línea en el cual apareció el error. Si el error sucedió en modo directo, ERL contendrá 65535.

ERR

Al producirse un error, ERR toma el valor del código de error correspondiente.

ERRORx

Se utiliza para forzar el error especificado. El entero X debe estar comprendido entre 1 y 255. BASIC utiliza los errores desde 1 a 59, los cuales, si son forzados, imprimen el mensaje de error respectivo.

El mensaje "Unprintable error" (error no imprimible) se produce si se fuerza un error utilizando la instrucción ON ERROR GOTO, para la cual no se ha hecho definición.

Ejemplo:

```
10 ON ERROR GOTO 200
.
.
100 ERROR 100
.
.
200 IF ERR = 100 THEN END
```

EXP(Y)

Calcula el valor e^y , donde e es la constante neperiana. La expresión Y debe estar comprendida entre -147.3654459516 y 145.06286085862

FIX(Y)

Entrega el valor entero de la expresión Y. Esta función difiere de INT en el tratamiento de los valores negativos, mientras FIX solamente trunca la parte fraccionaria del valor, INT redondea el valor hacia abajo.

Ejemplo: FIX (-2.8) da -2
INT (-2.8) da -3

FOR Y=x1 TO x2 STEP x3

Todas las instruccines siguientes hasta alcanzar la sentencia NEXT son repetidas con Y tomando valores desde x1 hasta x2, inclusive, en incrementos de x3. Si se omite STEP x3, el tamaño del incremento por omisión será 1. Si se fija en 0 el bucle es repetido continuamente. Si x2 es menor o igual a x1, siendo el incremento positivo, el bucle se ejecuta solamente una vez.

FRE ("")

Proporciona el número de bytes restantes, disponibles para almacenamiento de cadenas.

FRE(x)

Es una función que permite obtener el número de bytes disponibles para almacenamiento de programa Puede darse cualquier valor a x, puesto que es un argumento fantasma, y por tanto, no es utilizado en la evaluacion.

Ejemplo: Z = FRE(2)

GOSUB / RETURN

Variante: RETURN número de línea

Es una forma especial de la sentencia GOTO. Al encontrarse la instrucción RETURN, la ejecución del programa continua en la siguiente a GOSUB

GOTO

Esta sentencia provoca la bifurcación a la línea indicada.

HEX$(Y)

Esta función proporciona el valor hexadecimal equivalente al entero Y. Este último debe encontrarse entre -32768 y 65535

Ejemplo: HEX$(20) da 14.

IF condición THEN

Variantes: IF ... THEN ... ELSE
IF ... THEN ... IF ... THEN ... ELSE
IF ... GOTO
IF ... GOTO ... ELSE

Permite que determinadas sentencias sean ejecutadas, si se satisfacen ciertos criterios. Si se va a realizar una bifurcación, no es necesario utilizar GOTO después de THEN y ELSE. La segunda variante continuará la ejecución del programa a partir de la línea siguiente, si la condición inicial no es verdadera.

Ejemplo: 10 X=0:IF X=1 THEN IF Y=2 THEN 40 ELSE 80 continuará la ejecución en la línea 20

INP (dirección del puerto)

Lee un único byte de datos desde el puerto especificado.

INPUT Y

Variante: INPUT "mensaje"; Y
INPUT #número de archivo, Y
INPUT$ (x)
INPUT$ (x,número de archivo)

Esta sentencia imprime un signo de interrogacion para pedir al usuario una entrada de datos. Estos son asignados a las variables especificadas. Si la variable ha sido asignada antes de la sentencia INPUT y el usuario no entra datos antes de pulsar la tecla RETURN, el valor queda inalterado; en caso contrario, se asigna el valor 0.

Las variables pueden ser simples o de conjunto, numéricas o de cadena. Puede utilizarse más de una variable, separándolas con comas, tanto en la sentencia como en la entrada de datos.

Ejemplo: INPUT "X,Y$ (ENTER)";X,Y$ imprimirá X,Y$ (ENTER)?

La segunda variante es similar, con la excepción de que se especifica un número de archivo como fuente de datos. Las variantes 3 y 4 proporcionan una cadena de x caracteres, la variante 3 desde el teclado, y la variante 4 desde el archivo especificado. En ambos casos no es necesario finalizar la entrada pulsando RETURN.

INKEY$

Esta función entrega el primer carácter en la memoria intermedia de teclado, o si ésta está vacía, la cadena nula.

Ejemplo: 10 A$=INKEY$:IF A$="" THEN 10
Esto mismo puede conseguirse con mayor eficacia haciendo A$=INPUT$(1)

INSTR(A$,B$)

Variante: INSTR(x,A$,B$)

Entrega la posición de B$ en A$. Si B$ es la cadena vacía o no esta contenida en A$, se obtiene el valor 0. La variante permite que la búsqueda comience a partir de x caracteres a la izquierda de A$.

Ejemplo: A=INSTR(2),"ABCD","D") asigna a A el valor 4.

INT(Y)

Calcula el entero de Y. Veáse la instrucción FIX por su similitud.

Ejemplo: INT(4.8) da como resultado 4.

INTERVAL ON / OFF / STOP

Activa, desactiva o suspende, la llamada a una subrutina durante el intervalo especificado mediante la sentencia ON INTERVAL. Veáse el capítulo 2, para más detalles.

KEY número de tecla de función, "cadena"

Asigna la cadena dada, a la tecla de función especificada. La cadena puede tener hasta 15 caracteres de longitud

Ejemplo: KEY 2,"RUN"+CHR$(13)

KEY LIST

Produce el listado ordenado de las cadenas asignadas a las teclas de función.

KEY ON / OFF

Conecta o desconecta la visualización de las teclas de función que aparece en la linea vigèsimocuarta de la pantalla.

KEY (número de tecla de función) ON / OFF / STOP

Activa, desactiva o suspende, la llamada a una subrutina determinada por la sentencia ON KEY GOSUB. Si está activada, se realiza una comprobacion para ver si la tecla ha sido pulsada al terminar de ejecutarse cada sentencia de BASIC. Vease el capitulo 2 para más detalles.

LEFT$(Y$,x)

Devuelve x caracteres de Y$, empezando por la izquierda. Un carácter gráfico ocupa dos posiciones. Si x es 0, entrega la cadena nula. Si x excede del numero de caracteres de Y$, solamente se obtiene Y$, no se añaden espacios.

Ejemplo: PRINT LEFT$("NUEVO",2) imprimirá NU

LEN(Y$)

Esta función calcula el número de caracteres -incluyendo caracteres gráficos y de control- de Y$. Un carácter gráfico está compuesto de CHR$(1) más el código de carácter.

LET variable=expresión

Se utiliza para asignar el valor de la expresión a la variable, siendo opcional en el BASIC MSX.

LINE (x1,y1)-(x2,y2)

Variantes: LINE (x1,y1)-(x2,y2), color
LINE (x1,y1)-(x2,y2), color, B
LINE (x1,y1)-(x2,y2), color, BF
LINE -(x2,y2), color

La forma STEP(X,Y) puede reemplazar cualquiera de las especificaciones de coordenadas absolutas. En ambas posiciones, STEP hace relación al punto inicial de referencia. Esta instrucción dibuja en modo gráfico, una linea entre las coordenadas fijadas. La opcion B dibuja un rectángulo, y la opción BF lo dibuja y rellena de color. Veàse el capitulo 2 para más detalles.

LINE INPUT Y$

Variante: LINE INPUT "mensaje";Y$

Obtiene la entrada de una línea desde el teclado. Esta instruccion no puede usarse en los modos graficos. El número de caracteres escrito, no debe exceder de 254, asignándose su entrada al pulsar la tecla RETURN.

LINE INPUT# número de archivo, Y$

Recibe y asigna una linea (hasta 254 caracteres) desde el archivo secuencial especificado a la cadena dada.

LIST

Variantes: LIST número de línea
LIST número de línea-
LIST número de línea-número de línea
LIST -número de línea
LIST.

Esta instrucción permite obtener un listado del total, o de parte de un programa. Puede añadirse un punto (.) para listar:

1. La última línea listada anteriormente.

2. La linea que provocó la interrupción de la ejecucion del programa.

LLIST

Variantes: Las mismas que para LIST

LLIST envia el listado total o parcial del programa a la impresora.

LOAD "nombre de archivo"

Variantes: LOAD "referencia de dispositivo"
LOAD "referencia de dispositivo:nombre de archivo"
LOAD "referencia de dispositivo:nombre de archivo",R

La instrucción LOAD cierra todos los archivos abiertos, y borra el programa existente en memoria, antes de cargar el archivo ASCII determinado. Si se utiliza la opción R, no se cierran los archivos, y comienza la ejecucion al finalizar el proceso de carga Veáse la sentencia OPEN para obtener las referencias de dispositivos.

LOCATE columna,fila

Variantes: LOCATE ,fila
LOCATE ,,conexion de cursor
LOCATE columna,fila,conexión de cursor

La sentencia LOCATE se utiliza, solamente, en modo de texto, y desplaza el cursor a la columna y fila fijadas.

Modo de texto 40 columnas:

Columnas 0-36
Filas 0-22/23 (dependiendo de que se visualicen las teclas de función).

Modo de texto 32 columnas:

Columnas 0-28
Filas 0-22/23 (dependiendo de que se visualicen las teclas de función).

Las columnas no utilizadas pueden escribirse situando en la VRAM los valores adecuados (mediante VPOKE). Veáse el capitulo 2, para más detalles. La conexión del cursor puede tomar uno de los dos siguientes valores:

0: desconecta la visualización del cursor
1: visualiza el cursor

LOG(Y)

Calcula el logaritmo natural de la expresión Y. Esta expresión debe ser mayor que cero.

MAXFILES=Y

Determina el número máximo de archivos que pueden estar abiertos simultaneamente. La expresión entera Y estará comprendida entre 0 y 15. Esta sentencia es necesaria para abrir más de un archivo, a la vez.

MERGE

Variantes: MERGE "nombre de archivo"
MERGE "referencia de dispositivo"
MERGE "referencia de dispositivo,nombre de archivo"

Esta instrucción cargará y combinará el primer programa en ASCII que encuentre en la cinta con el programa existente en memoria. Si los números de líneas de ambos programas se superponen, las líneas del programa inicial se reemplazan por el segundo.

Ejemplo: Para cargar y mezclar el próximo programa en cinta, usaremos: MERGE "CAS:"

MID$(A$,x) = carácter

Variante: MID(A$,x,n) = n caracteres

Mediante esta instrucción se sustituye el x-ésimo carácter de A$. Todos los caracteres gráficos estan precedidos por CHR$(1). La variante permite reemplazar una secuencia de caracteres. Se obtiene una cadena vacia si x es mayor que la longitud de A$.

Ejemplo: A$="MAS":MID$(A$,1,2)=" E":?A$ imprime " ES"

MOTOR

Variantes: MOTOR ON (por omisión)
MOTOR OFF

Si no se especifica estado (ON conectado, OFF desconectado), se cambia el estado existente.

Ejemplo: Si el motor del cassette está desconectado, la instrucción MOTOR lo cambiará a conectado.

NEW

Borra el programa existente en memoria, y anula todas las variables.

OCT$(Y)

Entrega una cadena con el equivalente octal de la expresion decimal Y.

ON Y GOTO x1,x2, ... xn / ON Y GOSUB x1,x2, ... xn

Si Y=1 se produce un salto o llamada a subrutina, a la primera línea de la lista (x1). Si Y=2 el salto o llamada se realiza a la segunda línea y así, sucesivamente. Si la expresión Y=0 ó es mayor que el número de líneas incluidas en la lista, la ejecución del programa continua en la siguiente línea.

ON ERROR GOTO x

Si se produce un error, se fuerza el salto a la línea x. Si se especifica la línea 0, se vuelve al procedimiento usual de tratamiento de errores, imprimiéndose un mensaje de error. La rutina de tratamiento de errores finaliza con la instrucción RESUME.

ON INTERVAL = Y GOSUB X

Define la subrutina a ejecutar al finalizar cada intervalo (su longitud será Y*1/50 seg.). La secuencia comienza con la sentencia INTERVAL ON.

ON KEY GOSUB x1,x2, .. xn

Realiza una llamada a la subrutina correspondiente al pulsar la tecla de función respectiva. El primer número de línea corresponde a la tecla 1,

el segundo a la tecla 2, etc. La llamada se produce si la tecla de función se pulsa despues de una sentencia KEY(x) ON.

Ejemplo: ON KEY GOSUB 20,,,40 las teclas 2 y 3 no quedan activadas.

ON SPRITE GOSUB Y

Determina la subrutina que se ejecutará después de un choque entre sprites. Se activa mediante la sentencia SPRITE ON.

ON STOP GOSUB Y

Inicia una llamada a la subrutina de la línea Y, si se pulsan simultáneamente CTRL y STOP. Dicha llamada se produce solamente después de haber sido utilizada la sentencia STOP ON.

ON STRIG GOSUB x

Variantes: ON STRIG GOSUB x1,x2,x3,x4

Identifica la subrutina a ejecutar tras pulsar la barra de espacio. Su activación se realiza mediante STRIG ON (Y).

La variante se utiliza para permitir el empleo de los disparadores 1 y 2 de ambos mandos de juegos. Los números de las subrutinas deben escribirse en el siguiente orden:

1. Barra de espacio
2. Disparador 1, mando de juegos 1
3. Disparador 1, mando de juegos 2
4. Disparador 2, mando de juegos 1
5. Disparador 2, mando de juegos 2

Ejemplo: STRIG (2) ON:ON STRIG GOSUB 200,400,600 ejecutará una llamada a la subrutina de la línea 600, si se pulsa el disparador 1 del mando de juegos 2.

OPEN "referencia de dispositivo:" AS número de archivo

Variantes: OPEN "referencia de dispositivo:nombre de archivo" AS número de archivo

OPEN "referncia de dispositivo:nombre de archivo" FOR modo AS número de archivo

Esta sentencia abre un canal a un dispositivo, y le asigna una memoria intermedia de E/S. El archivo debe abrirse antes de poder utilizar cualquier instrucción que contenga un número de archivo. Por ejemplo, PRINT # o INPUT #. Pueden utilizarse las cuatro referencias de dispositivo siguientes:

1. CAS: Cassette
2. LPT: Impresora
3. CRT: Pantalla de texto
4. GRP: Pantalla de gráficos

El número de archivo se emplea por otras instrucciones de Entrada/Salida para hacer referencia al archivo, y debe estar incluido entre 0-Y, donde Y es el valor especificado en la instrucción MAXFILES.

La variante FOR modo, fija el tipo de transferencia de datos.

INPUT : Entrada secuencial
OUTPUT : Salida secuencial
APPEND : Apéndice secuencial

OUT número de puerto, byte de datos

Transfiere un byte de datos al puerto especificado. Tanto el número de puerto, como el dato, deben estar incluidos en el intervalo entero 0-255. La configuración MSX standard no admite más de 256 puertos de E/S.

PAD(Y)

Proporciona el estado de un tablero digitalizador. Y debe estar incluida entre 0 y 7:

0-3 Para tablero digitalizador conectado en el puerto 1
4-7 Para tablero digitalizador conectado en el puerto 2

Para cada uno de los cuatro valores que pueden seleccionarse por puerto, (0-3 y 4-7) se obtiene un parámetro diferente:

0 y 4: Toman el valor -1 si se presiona el tablero, y 0 si se libera
1 y 5: Proporciona la coordenada X del punto presionado
2 y 6: Proporciona la coordenada Y del punto presionado
3 y 7: Toman el valor -1 si se presiona el pulsador, y 0 si se libera

PAINT (X,Y)

Variantes: PAINT STEP (X,Y)
PAINT (X,Y), color
PAINT (X,Y), color, línea de borde

Esta instrucción puede usarse en los dos modos gráficos. Un objeto cerrado se rellena con el color de primer plano, desde la posición especificada. En el modo HRG (gráficos de alta resolución), el color de la línea de borde debe ser el mismo que el de la pintura. Veáse el capítulo 2, para más detalles.

PDL(Y)

Proporciona el estado de la raqueta conectada al terminal A ó B. La expresión Y será entera e incluida entre 1 y 12. Si su valor es impar, la entrada se lee desde el terminal A, y si es par, desde el B. El valor obtenido estará comprendido entre 0 y 255.

PEEK(Y)

Entrega el valor entero almacenado en la posición Y.

La expresión numérica Y debe estar comprendida entre 32768 y 65535.

PLAY "cadena de instrucciones"

Variantes: PLAY "cadena", "cadena", "cadena"
PLAY "cadena", "", "cadena"
PLAY Y$, Z$, X$
PLAY "Y$;cadena", "cadena", "cadena"

Es la principal instrucción para la generacion de sonido. Cada una de las tres cadenas especifica la salida de uno de los canales de sonido. Vease el capítulo 2, para más detalles.

PLAY (Y)

Esta funcion permite conocer el estado de los canales musicales. Y será entera, y comprendida entre 0 y 3. Si hay datos en la memoria intermedia del canal, se obtendrá el valor -1. Si el canal no está activado, se obtendrá el valor 0. PLAY (0) dá el estado de los tres canales.

POINT (X,Y)

Proporciona el código de color de un punto determinado de la pantalla. La instrucción opera solamente en los dos modos gráficos. Se obtiene el valor -1 si las coordenadas quedan fuera de los límites de pantalla.

POKE X,Y

Coloca el valor Y en la posición X. El valor Y estará entre 0 y 255, inclusive. El intervalo de direcciones abarcará desde -32768 a 65535. Una dirección negativa se sumará, previamente, a 65535.

Ejemplo: POKE 40000,254 situa el valor 254 en la dirección 40000

POS (Y)

Entrega la posición horizontal del cursor en cualquiera de los modos de texto. Y es un argumento fantasma, y no se emplea en la evaluación.

PRESET (X,Y)

Variantes: PRESET (X,Y), color
PRESET STEP (X,Y)

El punto de la pantalla de gráficos especificado, se colorea del color del fondo. La variante permite elegir color.

PRINT

Variantes: PRINT "cadena"
PRINT "cadena";

PRINT A$
PRINT Y
PRINT, X$
PRINT CHR$(Y)+A$

La instrucción PRINT puede abreviarse mediante el signo de interrogación (?). Si no hay ninguna expresión detrás de PRINT, se produce una alimentación de línea. Se realiza un retorno de carro despues de que el último artículo se haya imprimido, salvo que sea seguido por una coma o punto y coma.

Un punto y coma hace que el siguiente artículo se imprima inmediatamente despues del anterior, y una coma, avanza la posición de impresión a la próxima zona de tabulación. Las zonas de tabulacion tienen una anchura de 14 caracteres.

Los caracteres de control se imprimen mediante la función CHR$(Y).

PRINT # número de archivo, expresión

Es similar a PRINT, con la diferencia de que los datos se dirigen al archivo especificado. Un retorno de carro y una alimentación de línea siguen al articulo final.

PRINT USING

Variante: PRINT # numero de archivo, USING

Permite imprimir los diferentes artículos en un formato especificado, mediante el uso de caracteres de control. Veáse el capítulo 2, para más detalles.

PSET (X,Y), color

Variante: PSET STEP(X,Y), color

El punto de la pantalla de graficos de coordenadas (X,Y) se colorea del color fijado.

PUT SPRITE número de plano de sprite

Variantes: PUT SPRITE número de plano, (X,Y)
PUT SPRITE número de plano, STEP(x,y)
PUT SPRITE número de plano, (X,Y), color
PUT SPRITE número de plano, (X,Y), color, número de patrón

Es la principal instrucción para la visualización de sprites. PUT SPRITE fija la posición, color y patrón de un sprite. Las coordenadas referenciadas corresponden a la esquina superior izquierda del sprite.

Sólo puede situarse un sprite en cada plano, pero varios sprites pueden tomar un patrón determinado. Veáse el capítulo 2 y la instrucción SPRITE$ para más detalles.

Ejemplo: PUT SPRITE 0, (40,40), 4, 2

READ Y

Variante: READ Y,X,Z, ...

La sentencia READ se emplea para asignar las constantes contenidas en DATA a las variables de READ. Los datos asignados deben ser del mismo tipo que las variables.

REM

Sirve para escribir comentarios en un listado de programa Cualquier sentencia, incluida en la misma línea de programa, es ignorada en el momento de la ejecución, continuando la misma en la línea siguiente. REM puede abreviarse con el signo de comillas simples (')

RENUM

Variantes: RENUM nuevo número
RENUM nuevo número, viejo número
RENUM nuevo número, viejo número, incremento
RENUM viejo número de línea

El valor asumido por omisión para los parámetros nuevo número e incremento es de 10. Todas las líneas de programa son renumeradas, incluyendo los números de referencia que siguen a: GOTO, GOSUB, THEN, ELSE, ON ... GOTO, ON ... GOSUB, IF THEN y ERL.

El programa resultante comienza con el número de línea 10 (ó si se especificó, con el parámetro nuevo número). La variante "viejo número" hace que sólo parte del programa -aquella situada a continuación de viejo número- sea renumerada.

RESTORE

Variante: RESTORE número de línea

Sitúa el puntero de datos en el primer elemento de la sentencia DATA inicial. La variante permite especificar la línea de la sentencia DATA deseada.

Ejemplo: RESTORE 200 hará que la próxima sentencia READ tome los datos de la sentencia DATA situada en la línea 200 ó siguientes

RESUME

Variantes: RESUME NEXT
RESUME número de línea

La instrucción RESUME se emplea para finalizar una rutina de tratamiento de errores, en la que se entra, mediante la sentencia ON ERROR GOTO.

Hace que la ejecución continúe en la sentencia que produjo el error. La sentencia RESUME NEXT es similar, excepto que la ejecución prosigue en la sentencia situada a continuación de la que motivó el error original.

RIGHT$ (A$,x)

Esta función permite obtener una subcadena consistente en los x caracteres situados a la derecha de A$.

Si x es mayor que el número de caracteres de A$, se obtiene A$. Si x=0, se consigue una cadena nula. Un carácter gráfico incluye, como prefijo, CHR$(1)

Ejemplo: RIGHT$ ("IZQUIERDA",2) entrega DA

RND(Y)

Calcula un número aleatorio comprendido entre 0 y 1, ambos exclusive.

Si Y es mayor que 0, se genera el siguiente número de la secuencia.
Si es igual a 0, el número es igual al anterior.
Si Y es menor que 0, el generador de azar empieza de nuevo utilizando Y como dato de partida.

Es de destacar, que la secuencia producida después de hacer RND(-2) es la misma que la que volvería a obtenerse, haciendo de nuevo, RND(-2) (generador seudoaleatorio). Para generar una secuencia "más aleatoria", puede utilizarse RND(-TIME)

RUN numero de línea

Ejecuta el programa, existente en memoria, a partir de la línea especificada. Si no se escribe numero de linea, la ejecución comienza con la primera línea de programa.

SAVE "nombre de archivo"

Variantes: SAVE "referencia de dispositivo:"
SAVE "referencia de dispositivo:nombre de archivo"

Da salida a un archivo de programa BASIC en formato ASCII hacia el dispositivo especificado. CTRL Z equivale a EOF. No es posible verificar este tipo de archivos. La velocidad de transferencia es la indicada en la sentencia SCREEN.

Ejemplo: SAVE "CAS:PROG4"

SCREEN modo, tamaño de sprite, conexión eco de teclado, velocidad de transferencia, opción de impresora.

La sentencia SCREEN se utiliza para asignar las seis opciones de la forma siguiente:

Modo:

0	modo de texto de 40*24
1	modo de texto de 32*24
2	modo gráfico de alta resolución
3	modo gráfico multicolor

Tamaño de sprite:

0	8*8 puntos, sin ampliar
1	8*8 puntos, ampliado
2	16*16 puntos, sin ampliar
3	16*16 puntos, ampliados

Conexión eco de teclado:

0	eco conectado
1	eco desconectado

Velocidad de transferencia (cassette):

1	1200 baudios
2	2400 baudios (para los dos formatos)

Opción de impresora:

0	impresora MSX
≠ 0	impresora no MSX

Si se adopta la última opción, los símbolos gráficos se envían a la impresora en forma de espacios.

SGN (Y)

Si Y es mayor que 0, SGN entrega 1. Si Y = 0, entrega 0, y -1, en los restantes casos.

SIN (Y)

SIN calcula el seno de la expresión Y, dada en radianes.

SOUND registro, Y

La instrucción SOUND escribe el valor de la expresión Y en el registro especificado del PSG (generador programable de sonido). Y debe estar comprendido entre 0 y 255. Veáse el capítulo sobre el AY-3-8910 para más detalles.

Ejemplo: SOUND 4,4

SPACE$ (Y)

Genera una cadena de Y espacios. Y debe estar comprendida entre 0 y 255.

SPC (Y)

Se utiliza, en unión de PRINT ó LPRINT, para producir Y espacios. Y estará entre 0 y 255, ambos inclusive.

Ejemplo: PRINT SPC(4); "TITULO"

SPRITE$ (Y) = "cadena de definición"

Es una variable del sistema para contener la definición de un patrón de sprite. El código de carácter de los caracteres asignados a la cadena, es el patrón de un byte del sprite. La forma binaria permite una entrada directa de dicho byte, haciendo:

CHR$(&B10101011)

si se utlizan sprites de 16 puntos, pueden definirse hasta 64 patrones de sprite, e Y debe estar comprendida entre 0 y 63. Los sprites de 8 puntos permiten hasta 255 definiciones. Veáse el capítulo 2, para más detalles.

SPRITE ON / OFF / STOP

Activa, desactiva o suspende, la llamada a una subrutina en el caso de que se produzca una colisión entre sprites. La sentencia ON SPRITE GOSUB debe haber sido utilizada para especificar la llamada a la subrutina.

Ejemplo: SPRITE STOP suspenderá las llamadas a la subrutina al producirse un choque entre sprites. La llamada se produce cuando la detección ha sido activada mediante una sentencia SPRITE ON.

SQU (Y)

Calcula la raiz cuadrada de la expresión Y, que debe ser igual o mayor que 0.

STICK (Y)

Y puede tomar los valores desde 0 a 2:

0 Teclas de cursor
1 Mando de juegos en el puerto 1
2 Mando de juegos en el puerto 2

El valor obtenido depende de la dirección, según el esquema siguiente:

```
                1
      8         |         2
 7    -         0         -         3
      6         |         4
                5
```

STOP

Esta sentencia finaliza la ejecución del programa, e imprime el mensaje:

Break in "numero de línea"

STOP ON / OFF / STOP

Activa, desactiva o suspende, la llamada a la subrutina especificada en el caso de que hayan sido pulsadas las teclas CTRL y STOP. Veáse el capítulo 2, para mas detalles.

STRIG (Y)

La expresión entera Y debe estar comprendida entre 0 y 4, con las siguientes funciones:

0 Barra de espacio
1 Mando de juegos 1, disparador 1
2 Mando de juegos 2, disparador 1
3 Mando de juegos 1, disparador 2
4 Mando de juegos 2, disparador 2

Si se pulsa el disparador, se produce el valor -1, y 0, en caso contrario.

STRIG (Y) ON / OFF / STOP

Activa, desactiva o suspende, la detección de un disparador determinado. Y estará entre 0 y 4, segun las funciones expresadas anteriormente. La sentencia ON STRIG GOSUB debe utilizarse previamente para definir la subrutina a ejecutar.

Ejemplo: STRIG (0) ON activa la detección al pulsar la barra de espacio.

STRING$ (x,Y)

Variante: STRING$ (x,Y$)

Produce una cadena de longitud x, conteniendo un único carácter cuyo código es Y ó el primer carácter de Y$.

Si el primer carácter es gráfico, se obtiene una cadena formada por caracteres 1.

Ejemplo: STRING$ (4,65) dá como resultado AAAA

STR$ (Y)

Entrega una cadena, representando el valor numérico Y. Se tiene, así, un procedimiento de conversión para las formas hexadecimales, octales o en notación científica.

SWAP Y,X

Intercambia los valores asignados a cada variable. Ambas variables deben ser del mismo tipo.

Ejemplo: A$ = "W":B$ = "E":SWAP A$,B$:PRINT A$ imprime E

TAB (Y)

Esta función puede usarse solamente detrás de las sentencias PRINT ó LPRINT TAB desplaza la posicion de impresión a la columna Y -a no ser que se encuentre a la derecha de dicha columna, en cuyo caso, no sucede nada-. Y debe estar comprendida entre 0 y 255.

Ejemplo: PRINT TAB(4);"EN LA CUARTA COLUMNA"

TAN (Y)

Calcula la tangente de la expresión Y, dada en radianes.

TIME

Es una variable entera, especial, que toma el valor 0 al conectar el ordenador, y se incrementa cada 1/50 seg. Este aumento se produce durante la interrupción del VDP, que se suspende en la entrada o salida de datos hacia el cassette.

TRON / TROFF

La sentencia TRON hace que se imprima el número de línea de cada sentencia del programa al ejecutarse ésta. La sentencia TRON puede utilizarse en los modos directo e indirecto, y se desconecta mediante TROFF.

Ejemplo:

```
10 TRON
20 REM
30 REM
40 TROFF
```

imprimirá (20)(30)(40)

Z=USR (Y)

Variante: Z=USR x (Y)

USR x (Y) hace una llamada a la dirección especificada mediante la instrucción DEFUSR x. El valor de la expresión Y se pasa a la rutina. Si no se precisa transferir ningún argumento, debe darse un valor fantasma a Y.

Al completarse la rutina en lenguaje máquina, el valor obtenido es pasado y asignado a Z.

Tanto en la llamada a subrutina, como al retornar, los datos que integran el valor que se intercambia, se almacenan a partir de &HF7F6 El tipo de datos se indica mediante el valor del acumulador:

Entero	2
Cadena	3
Precisión sencilla	4
Precisión doble	8

La ubicación del descriptor de cadena, se transfiere mediante el par de registros DE. Este descriptor está formado por tres bytes:

1. El número de caracteres de la cadena.
2 y 3. La dirección de almacenamiento de la cadena.

Ejemplo:
```
10 DEFUSR4=&HD000
20 Y=USR 4 (X)
```

VAL (Y$)

Calcula el valor numérico de la cadena Y$. Los espacios previos, tabulaciones o alimentación de líneas se ignoran.

VARPTR (Y)

Variante: VARPTR (# número de archivo)

Esta fución calcula la dirección inicial de la tabla de datos de la variable especificada.

Puede emplearse con cualquier tipo de variable. Si la dirección obtenida es negativa, debe sumarse 65536 para conseguir la dirección correcta. Veáse la sección "Almacenamiento de programa", al final del capítulo 2, para más detalles.

La variante VARPTR (# número de archivo) calcula la dirección inicial del bloque de control de archivo.

VDP (Y)

Es una variable especial, donde Y debe estar comprendida entre 0 y 8, ambos inclusive. Los valores 0 a 7 entregan el valor almacenado en

el registro de escritura del VDP asociado a dichos números. VDP(8) proporciona el valor del registro de estado del VDP. Veáse la sección sobre el VDP, para más detalles

VPEEK (dirección)

Entrega el valor almacenado en la dirección especificada de la VRAM. Dicha dirección debe estar comprendida entre 0 y 16383, ambos inclusive.

VPOKE X,Y

Esta función sitúa el valor Y en la dirección X de la VRAM Y debe estar comprendido entre 0 y 255, ambos inclusive.

WAIT puerto, valor de byte

Variante: WAIT puerto, valor de byte, máscara

Esta instrucción detiene la ejecución del programa hasta que se produzca a entrada de un byte por el puerto especificado. El byte debe tener uno de sus bits a nivel 1, estando dicho bit en la misma posición que uno de los bits del byte especificado en la sentencia.

La variante opera, en primer lugar, el byte de entrada con el byte máscara, mediante el operador lógico OR. A continuacion, el resultado es operado, de nuevo, mediante el operador lógico AND, con el valor de byte.

Todas las expresiones deben ser enteras, y comprendidas entre 0 y 255, inclusive

WIDTH anchura de la pantalla de textos

Esta instrucción fija la anchura (en columnas) de la pantalla, en cualquiera de los modos de texto.

El valor asignado debe estar incluido en los siguientes intervalos:

1-40 en el modo de texto de 40*24
1-32 en el modo de texto de 32*24

Ejemplo: WIDTH 20

CAPITULO 4

Lenguaje Máquina Z-80

Microprocesadores. La Arquitectura del Z-80.

Juego de Instrucciones. Direccionamiento.

MICROPROCESADORES

En el corazón de cada ordenador del sistema MSX se encuentra un microprocesador Z-80A. Esta unidad se programa directamente utilizando código máquina, y tiene, en principio, tres capacidades:

a) Aritmética sencilla.

b) Movimiento de datos entre distintos lugares.

c) Operaciones lógicas.

Un programa en código máquina, llamado intérprete de BASIC, se encuentra en ROM (memoria de sólo lectura). Este programa traduce las instrucciones en BASIC a instrucciones en código máquina, que son ejecutadas por el Z-80.

La omisión de esta fase de traducción, permite que un programa en código máquina se ejecute más rápidamente que su equivalente en BASIC. De aqui proviene la característica de BASIC de ser un "lenguaje de alto nivel", que admite órdenes que no son intrínsecas del microprocesador.

Así que, la velocidad es la principal ventaja de utilizar lenguaje de máquina, ¿pero, cuáles son los inconvenientes? Mientras que BASIC fue diseñado para permitir que el ordenador sea usado y programado con muy poco conocimiento de la máquina, el trabajo a nivel de lenguaje de máquina requiere un poco más de atención a lo que está sucediendo y familiaridad con los sistemas de numeración binario y hexadecimal. Un pequeño inconveniente, a cambio de los deslumbrantes efectos que pueden ser realizados.

Hasta ahora sabemos que existe un Z-80 en el interior del ordenador apoyado por un 9129 VDP (procesador de pantalla de video) y un AY-3-8910 (circuito de sonido), ¿pero cómo se conecta todo esto?

La comunicación entre las unidades se realiza a través de dos caminos principales: el bus de direcciones y el bus de datos. El primero de ellos está formado por 16 líneas o hilos, que conectan el Z-80 y los bloques de memoria ROM y RAM (memoria de acceso aleatorio). El bus de datos lleva un camino paralelo, pero sólo consta de 8 líneas.

Utilizando el bus de direcciones de 16 líneas, el Z-80 puede especificar cualquier posición en el intervalo comprendido entre 0 y 65535. Cada posición, sea de ROM o de RAM, puede contener un número entre 0 y 255.

Cuando el Z-80 envía una dirección al bus, e indica a la memoria que se necesita una lectura, y no se requiere una escritura, el número almacenado con esa direccion es, automáticamente, colocado en el bus de datos El procesador, entonces, le da entrada en uno de sus registros internos.

Es preciso que el sistema esté exactamente sincronizado, para ello se utiliza una referencia de tiempo, un reloj funcionando a una frecuencia de 3.75 MHz. Este reloj es externo al Z-80.

El VDP también tienen acceso al bus de datos, para permitir que la CPU (unidad central de proceso) le envíe datos e instrucciones. Adicionalmente, el VDP tiene 16K de RAM, a la que sólo él puede acceder directamente. Para ello, dispone de un bus de datos, bidireccional, de 8 bits.

Mediante las instrucciones de modo dadas por el procesador, y los contenidos de la VRAM (memoria de acceso aleatorio para video), el VDP construye una pantalla de video. Esta señal es enviada, directamente, a un monitor, o modulada para verse en un televisor.

Al conectar el ordenador, las definiciones de los juegos de caracteres que se encuentran en ROM, son transferidas por la CPU al VDP, y de aquí, a la VRAM.

ORGANIZACION DEL SISTEMA

Cuando se trabaja con áreas de memoria de gran tamaño, es conveniente dividirlas en bloques más manejables. Una división común es mediante páginas de 256 direcciones. La página inicial se conoce como página 0, y contiene las posiciones desde 0 a 255. El término página también puede usarse para hacer referencia a bloques de memoria de 16K. El tiempo que necesita la CPU para transferir datos con la página 0, es menor que cuando hay que atravesar un límite de página. En consecuencia, la mayoria de los pequeños ordenadores, reservan la página 0 para uso del sistema.

Las instrucciones del microprocesador tienen una longitud de 1 ó 2 bytes. Si además, hace falta especificar datos o una dirección, serán necesarios 1 ó 2 bytes adicionales.

Así, un programa en lenguaje máquina, adopta la forma de una secuencia continua de bytes de instrucciones y datos.

Algunas posiciones de memoria internas del procesador, son accesibles para el programador. Las más útiles son las siguientes:

1. Contador de programa.
2. Puntero de pila.

3. Dos registros de índice: IY e IX
4. El acumulador o registro A
5. Seis registros generales, del B al L
6. El registro de banderas F
7. Los registros de interrupción y refresco I y R
8. Un juego de registros alternativos desde A' a L'

Los registros desde el 1 al 3, tienen 16 bits de longitud, y pueden contener cualquier número comprendido entre 0 y 65535. El resto, tienen 8 bits de longitud, y se limitan a valores entre 0 y 255, ambos inclusive.

En primer lugar, el procesador debe saber dónde se encuentran sus instrucciones, siendo ésta la función del registro de contador de programa, que contiene la dirección de la siguiente instrucción.

l registro de banderas contiene información sobre el estado del sistema, y los resultados de ciertas operaciones. Por ejemplo, si una resta obtiene un valor negativo, el bit de signo (S) se situará a nivel 1

El registro de banderas se emplea por las opciones de bifurcación incluidas en la lista de instrucciones. Estas, apuntan el contador de programa a una nueva posición, en función del nivel (1 ó 0) de una de las banderas.

Los dos registros de índice IY e IX pueden cargarse con cualquier valor que el programador desee -siempre que esten dentro del intervalo permitido-. Estos registros tienen como función principal contener la dirección de datos o rutinas.

Los registros generales: B, C, D, E, H, L, también pueden cargarse con cualquier valor del intervalo. Además, pueden aparearse -BC, DE, HL- para usarse como registros de 16 bits.

El registro A, o acumulador, tiene una importancia especial, que deriva de la arquitectura del Z-80. La mayoría de las operaciones aritméticas o lógicas de 8 bits, requieren que uno de los valores sea cargado inicialmente en el acumulador. Después de realizada la operación, el resultado es, asímismo, almacenado en el acumulador.

Aunque esto permite que dichas operaciones se ejecuten rápidamente, normalmente se requieren instrucciones adicionales para cargar y transferir datos desde el acumulador. El Z-80 también es capaz de efectuar adiciones y sustracciones de 16 bits. El par de registros HL reemplaza, en este caso, al registro A como acumulador. El juego de registro alternativo puede conectarse para sustituir los registros A, B, C, D, E, F, H, L, por A', B', C', D', E', F', H', L'. De esta forma, existen dos juegos de registros disponibles para un manejo/almacenamiento inmediato de datos. En la práctica, sin embargo, el segundo juego se usa escasamente, con el fin de que las interrupciones puedan ser rápidamente terminadas.

NOTACION BINARIA Y HEXADECIMAL

Antes de pasar a otro apartado, conteste a estas dos preguntas:

1. ¿Cuáles son las representaciones binarias de -34 y 0.02?
2. ¿Cuál es el equivalente decimal de &HOFDE?

Los microprocesadores no han sido diseñados para utilizar el sistema de numeración decimal, y para programar el lenguaje máquina, es necesario dominar los sistemas de numeración binario y hexadecimal.

En el sistema decimal existen dígitos 0-9, en binario hay solamente dos 0 y 1, por lo tanto, cualquier número binario es una secuencia de unos y ceros.

Otra diferencia principal consiste en que cada columna representa, en binario, un valor doble del de la columna situada a su derecha, mientras que en decimal, dicho valor es diez veces mayor

DECIMAL	NOTACION BINARIA								
	128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	1	0	
3	0	0	0	0	0	0	1	1	
4	0	0	0	0	0	1	0	0	
5	0	0	0	0	0	1	0	1	
6	0	0	0	0	0	1	1	0	
7	0	0	0	0	0	1	1	1	
8	0	0	0	0	1	0	0	0	= 8*1+4*0+2*0+1*1
9	0	0	0	0	1	0	0	1	
100	0	1	1	0	0	1	0	0	
200	1	1	0	0	1	0	0	0	= 128+64+8

Figura 4.1 Valores de numeración binaria

Para convertir un número binario en su equivalente decimal, es necesario sumar aquellos valores de las columnas en las cuales el número tiene unos. En la figura 4.1 el equivalente binario de 100 se muestra como 64+32+4. Cada uno o cero se llama bit. Un número que consta de 8 bits, se denomina byte.

Está claro que el mayor número que puede representarse con un número de 8 bits en binario, es el 128+64+32+16+8+4+2+1=255. Por convenio se incluyen ceros para completar por la izquierda los 8 bits para aquellos valores menores que 255.

El bit situado a la derecha es el menos significativo (l s b. en ingles) y el bit situado más a la izquierda es el bit más significativo (m.s.b). Estas abreviaturas no deben confundirse con M.S.B. (byte más significativo, en ingles) y L.S.B. (byte menos significativo). Por ejemplo, en una dirección de 16 bits como:

1111111100000010

el l.s.b. es 0, el m.s.b. es 1, y el M.S.B. es 1111111.

En los siguientes capítulos tendremos, con frecuencia, que referirnos, a un bit específico dentro de un byte. Convencionalmente, los bits se numeran de la siguiente forma:

76543210

La suma y resta binarias son directas, con una excepción, la representación de un numero negativo. Como no existe equivalente del signo menos (-) los numeros negativos se forman de diferente manera. El método de codificación empleado es el de la notación de complemento de 2. Para ello, se toma el valor absoluto del número negativo y se convierte a su forma binaria. A continuación, cada dígito es invertido (se convierte en 0 si es 1, y en 1 si es 0). Por último, se le suma 1.

Por ejemplo, para obtener el complemento de 2 de -12, tomamos la representación binaria de su valor absoluto (12) 00001100. Lo invertimos: 11110011, y después le sumamos 1, obteniendo 11110100. Con el objeto de que el procesador sea capaz de diferenciar este valor del equivalente binario de 244 decimal, que tiene la misma representación, se impone un límite al intervalo de valores: -128 a 127. Se evita, así, que pueda producirse, un solapamiento.

OPERACION	EJEMPLO
Valor negativo	-121
Valor absoluto	121 / 01111001
Forma invertida	10000110
Incremento	00000001
Complemento de 2	10000101

Figura 4.2 Representación de valores negativos en notación de complemento de 2.

La división por 2 de un valor binario se consigue moviendo cada bit una posicion a la derecha. Por lo tanto, para obtener el equivalente binario de 1/2, el byte 00000001 se desplaza, obteniéndose 0.1. La figura 4.3 muestra los valores equivalentes a partir del punto binario.

DECIMAL	FRACCION BINARIA
0.5	0.1
0.25	0.01
0.125	0.001
0.0625	0.0001
0.03125	0.00001

Figura 4.3 Equivalentes de fracciones binarias

Un número de 8 bits puede dividirse por la mitad, en dos partes de 4 bits, llamadas "nibbles". Cada una de ellas representa, aisladamente, un valor comprendido entre 0 y 15. Esta es la base de un atajo en el manejo de datos binarios. Está claro, que vamos a necesitar 16 dígitos diferentes, por lo que los valores 0-9 van a ser ampliados con las seis primeras letras del alfabeto, que van a representar los valores 10-15.

DECIMAL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
HEXADECIMAL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11

Figura 4.4 Notación hexadecimal

Este método de representación es el sistema de numeración hexadecimal. Por ejemplo, para convertir 56 decimal, a hexadecimal, tomamos el equivalente binario: 00111000 y extraemos el valor de cada nibble aislado: 0011 y 1000 = 3 y 8, respectivamente. El equivalente hexadecimal es, por tanto, 38.

También conseguimos mayor simplicidad, en relación con el sistema bianario. Para convertir un valor hexadecimal de dos dígitos a decimal, simplemente hay que multiplicar el dígito situado a la izquierda por 16, y sumarlo al valor de la derecha.

La notación hexadecimal es el sistema de numeración en base 16. Para convertir un valor decimal a hexadecimal, es necesario dividir el valor entre las potencias de 16 que lo componen.

HEXADECIMAL	4	3	A	4	
VALOR DE COLUMNA	4096	256	16	actual	
EVALUACION	4*4096+	3*256+	10*16+	4	= 17316

Figura 4.5 Conversión hexadecimal a notación decimal

OPERACIONES LOGICAS

Consideremos el siguiente problema:

Sean dos números binarios de 8 bits, se precisa formar otro byte con unos solamente en aquellas posiciones en que ambos números tengan unos.

La respuesta es simple, primero cargamos un valor en el registro A, después ejecutamos la operación lógica AND entre él y el otro byte

Al completarse, el resultado -el byte pedido- se encuentra en el registro A.

La operación AND es una de las tres operaciones lógicas que el Z-80 puede realizar. Dichas operaciones comparan cada uno de los bits de dos bytes, y en función del resultado de esa comparación, crean un nuevo valor que se almacena en el acumulador. Como hemos visto, la operación AND proporciona un cero en cada una de las posiciones de los bits, salvo que ambos datos contengan un uno en dichas posiciones. La operación inversa a ésta, se denomina OR, y situa un uno en cada posición si uno o ambos de los datos iniciales, contienen un uno La tercera instruccion, XOR, es un cruce entre las dos anteriores. Precisa que ambos datos tengan dígitos diferentes, para producir un uno como resultado en el acumulador.

Para el recien llegado al código máquina, estas instrucciones pueden parecer de poca importancia, sin embargo, no es así, son frecuentemente utilizadas para ciertos calculos y para el control de funciones particulares.

OPERACION	AND	XOR	OR
Condición para obtener 1 como resultado	Ambos datos tienen un 1 en columna	Ambos datos tienen valores diferentes en columna	Uno o ambos valores en columna son 1

Figura 4.6 Base de comparación de operaciones lógicas.

LA ARQUITECTURA DEL Z-80

Zilog ha desarrollado varias versiones del Z-80, que se diferencian en la maxima velocidad de funcionamiento. Para el Z-80A del sistema MSX; la máxima velocidad de reloj es de 4 MHz. (el Z-80B puede funcionar hasta 6 MHz.).

La configuración estructural del procesador es un encapsulado standard de 40 patillas, en doble fila (D.I.L.P.). El bus de direcciones consta de 16 líneas, y el bus de datos de 8 líneas. Las funciones de las restantes patillas son:

1. Alimentación.
2. Patillas de puesta a cero e interrupción: RESET, INT y NMI.
3. Detención del ordenador para permitir a la memoria la recuperación de los datos necesarios: WAIT.
4. Líneas de control para permitir que otros procesadores accedan a los buses de direcciones y datos: BUSREQ y BUSAK.
5 Indicador del refresco de memoria mediante las 7 líneas inferiores del bus de direcciones: RFSH.

6. Patillas de entrada de señal de reloj.
7. Patillas de lectura y escritura (RD y WR).
8. Patillas MREQ y IORQ.

Además de ser capaz de transferir datos hacia la memoria, el procesador puede recibir o enviar datos a 256 puertos de 8 bits. Los registros de control de los componentes fundamentales del sistema, se acceden de esta forma. De esta manera, la mayoría de las operaciones de E/S se inician por la CPU escribiendo a los puertos correspondientes.

Para conseguir la atencion del procesador, en orden a realizar una función urgente, uno de los componentes puede producir una interrupción. Al producirse ésta, el Z-80 finaliza la instrucción que esté realizando en ese momento y se dirige a una rutina de control. Una vez terminada, continua ejecutando el programa original.

Frecuentemente, no es necesario utilizar los procedimientos de interrupción, si solamente se requiere el uso de los buses de direcciones y datos. El Z-80 es capaz de aislarse asímismo de los buses de direcciones y datos. La línea BUSREQ requiere el uso de los buses desde la CPU, la cual reconoce su estado de aislamiento mediante la línea BUSAK.

La línea MREQ indica que el procesador va a realizar una lectura o escritura a memoria, y que ha colocado una dirección en el bus de direcciones. IORQ adquiere un nivel bajo para señalar que se ha producido una interrupción, o que se está procediendo a una operación de E/S.

Internamente, el Z-80 esta integrado por una unidad lógico-aritmética (ALU), una unidad de control, los registros y la memoria de sólo lectura (ROM). Su funcionamiento se describe mejor mediante un ejemplo de operación:

1. Se accede a la dirección señalada por el contador de programa, y su contenido se carga dentro del registro de instrucción del procesador. Supongamos que sea la 01111100 ó 124 en decimal, que hace que el procesador cargue el acumulador (registro A) con el contenido del registro H.

2. La CPU incrementa el contador de programa, para que apunte a la diguiente instruccion.

3. El byte del registro H se copia en el registro A, eliminando cualquier valor previo.

4. La siguiente instrucción se carga en el procesador ...

La operación se completa en 4 T ciclos (períodos de reloj externos). Esto da idea de la velocidad del sistema, y de por qué los errores de programación son tan difíciles de detectar. Aunque en comparación con la mayoría de los procesadores de 8 bits, el Z-80 está bien equipado de registros, éstos son, con frecuencia, inadecuados para almacenar toda la informacion que se precisa Afortunadamente, el Z-80 tiene un almacén mayor -la pila-.

La pila (stack) es una parte de memoria que puede utilizarse para almacenar datos de 2 bytes. Los datos almacenados deben proceder de cualquier registro de 16 bits, o de un par de registros, con la excepción del contador de programa.

La pila, como su nombre indica, va apilando los valores introducidos, uno encima del otro, con el inconveniente de que está boca abajo. En el momento que un par de bytes sean colocados sobre la pila, la parte "superior" de la misma desciende dos posiciones en la memoria. La posición del último byte del par recién colocado, queda recogida por el registro del puntero de pila.

Un inconveniente de la pila, es que los pares de bytes deben ser retirados según el proceso: último en entrar, primero en salir.

Ejemplo: Supongamos que el contenido del registro de 16 bits IY sea enviado a la pila, seguido por el valor del par de registros BC. Para retirar el primer par introducido, se hace necesario, sacar primero el par BC. La pila también se emplea para almacenar las direcciones de retorno de subrutinas e interrupciones.

Antes de presentar el conjunto de instrucciones del procesador, trataremos, brevemente, de los 256 puertos de E/S. Se puede leer o escibir un único byte de datos a cualquier puerto, empleando la instrucción correspondiente. El procesador utiliza las 8 líneas de direcciones de orden inferior A0-A7, para seleccionar el puerto (no se produce una lectura o escritura a memoria, puesto que es la línea IORQ y no la MERQ, la conectada). El byte de datos se sitúa en el bus de datos y se recibe por el Z-80 (lectura) o por el puerto (escritura).

EL CONJUNTO DE INSTRUCCIONES DEL Z-80

El Z-80 tiene 158 tipos de instrucciones, que Zilog agrupa en once categorias. Antes de considerarlas, se hace preciso estudiar los métodos principales para introducir o escribir programas en lenguaje máquina.

Una rápida mirada a las revistas especializadas de ordenadores, nos revela que hay dos métodos principales para documentar las rutinas en lenguaje máquina. La mas popular consiste en listar los valores hexadecimales que componen cada byte de la rutina. El usuario, sitúa estos valores en memoria, mediante la instrucción POKE, y después las ejecuta llamando a su primera dirección con la instrucción USR.

Estos bytes se denominan programa objeto o código objeto. En este formato la secuencia de bytes puede ser procesada directamente por el Z-80.

Sin embargo, situar cada byte en memoria, es una tarea laboriosa y sujeta a errores. Una solución sencilla consiste en escibir una corta rutina en BASIC que acepte números hexadecimales de dos dígitos, y vaya colocando su equivalecte decimal, en posiciones consecutivas de la memoria. Esto se conoce como cargador hexadecimal, del que se acompaña un ejemplo:


```
100 '******************************
110 '*****CARGADOR  HEXADECIMAL*****
120 '******************************
130 '  INTRODUZCA LA DIRECCION DE          COMIENZO EN DECIMA
L Y DESPUES
140 'EL CODIGO EN HEXADECIMAL, DE DOS     EN DOS DIGITOS.
150 ' PARA FINALIZAR LA INTRODUCCION      DE CODIGO ESCRIBA
FI.
160 'EL PROGRAMA PRESENTA EN PANTALLA     LAS DIRECCIONES Y
EL CODIGO EN
170 'DECIMAL Y HEXADECIMAL.
200 SCREEN 1:COLOR 10,1,1:CLS
210 LOCATE 2,2:PRINT "CARGADOR HEXADECIMAL, ESCRIBA FI PARA
TERMINAR":PRINT
220 INPUT "DIRECCION DE COMIENZO";ST
230 SW=ST-1:PRINT
240 SW=SW+1:PRINTSW;" ";HEX$(SW);" ??";
250 PRINT CHR$(29);CHR$(29);
260 A$=INPUT$(2)
270 IF A$="FI" OR A$="fi" THEN 320
280 A$(1)=MID$(A$,1,1):A$(2)=MID$(A$,2,1)
290 IF INSTR("0123456789ABCDEFabcdef",A$(1))=0 OR INSTR("012
3456789ABCDEFabcdef",A$(2))=0 THEN 260 ELSE PRINTA$;
300 A=VAL("&H"+A$):PRINT" ";A
310 POKE SW,A:PRINT:GOTO 240
320 PRINT:PRINT:PRINT:PRINT"PRINCIPIO ";ST;" FIN ";SW-1
```

Si el programa en lenguaje máquina es largo, el proceso de buscar los códigos de byte de cada instruccion es práctico, y se hace necesario utiiizar un ensamblador.

Un programa ensamblador permite que la rutina sea escrita empleando mnemónicos, para las instrucciones, y si es preciso, cadenas para las direcciones. El programa fuente se convierte en programa objeto mediante el ensamblador. Cuando se alcanza una cierta familiaridad con los principales mnemónicos, la producción de código máquina es casi tan simple, como escribir un programa en BASIC. El programa fuente consiste, normalmente, en una secuencia de líneas numeradas, cada una de las cuales contiene una o más instrucciones en lenguaje máquina. Todos los ensambladores de cierta calidad permiten utilizar el editor de pantalla, y emplean instrucciones de una sola letra.

La mayoria de los ensambladores comerciales constan de tres partes:

1. Ensamblador.
2. Monitor.
3. Desensamblador.

Una vez que le programa fuente está completo, se ensambla en código máquina en la dirección especificada. A continuación, el programa se almacena en cassette, puesto que puede contener errores que hagan

necesario desconectar el ordenador antes de que vuelva a poder utilizarse. Por último, se introduce el monitor para ejecutar el programa -si es preciso, de instrucción en instrucción-. El desensamblador se emplea para examinar y alterar las zonas de memoria, y por lo general, se accede a él desde el monitor.

Además de estas operaciones fundamentales, existe un conjunto de funciones secundarias, como son: mover, almacenar o cargar bloques de memoria, operaciones de búsqueda de cadenas, etc.

La aplicación elegida deberá determinarse por la tarea de programación que pretenda realizarse. El intervalo de precios es considerable. Para un uso escaso, será suficiente con un sistema basado en cinta, pero para aplicaciones más serias, será casi obligatorio un paquete basado en cartucho. La principal razón, para ello, es que la mayoria de los programas en lenguaje máquina contendrán, inicialmente errores, y el hecho de volver a cargar la aplicación desde el cassette después de cada error no recuperable, puede producir más de una irritación.

Aunque hay 158 tipos de instrucciones admitidas por el procesador, existen 666 códigos individuales. Esto se debe a que la mayoría de las categorias de instrucciones contienen un espectro de operaciones, cada una de las cuales ejecuta una función idéntica, pero con diferente formato de especificación de direcciones o datos.

El Z-80 permite 10 modos de direccionamiento, que examinaremos más de cerca en el próximo apartado. El conjunto de instrucciones puede dividirse en las siguientes categorías:

1. Operaciones de carga de 8 y 16 bits.
2. Operaciones aritméticas de 8 y 16 bits.
3. Instrucciones lógicas de 8 bits.
4. Operaciones de rotación y de traslado.
5. Instrucciones de manejo de bits.
6. Operaciones de bifurcación y subrutinas.
7. Operaciones de transferencia de bloques y de búsqueda.
8. Instrucciones de control de la CPU y de E/S.

OPERACIONES DE CARGA DE 8 Y 16 BITS

Una instrucción de carga de 8 bits copia el contenido de un registro o posición de memoria en otro registro o posición de memoria. Las operaciones de 16 bits realizan una función equivalente, entre los registros de 16 bits y la memoria u otros registros.

Cuando un valor de 16 bits se almacena desde un registro a una posición X, el byte de mayor orden se sitúa en la posición X+1, y el byte de orden menor en la posición X. En forma similar, si se carga un registro de 16 bits con los dos bytes de la posición X, el valor de la posicion X+1, se copia en los 8 bits más significativos del registro, y el valor de X en los 8 bits menos significativos.

El mnemónico de ensamblador Zilog de esta operación es LD X,Y donde Y es el byte copiado en la posición o registro X.

Ejemplos:

LD A,B carga el registro A con el contenido del registro B
LD H,A carga el registro H con el contenido del registro A
LD B,40 carga el registro B con el valor 40

Si un operando está encerrado entre paréntesis, hace referencia a la direccion del dato a utilizar.

LD A,(40) carga el acumulador con el valor almacenado en la posición 40

De forma semejante, LD B,(HL) cargará el registro B con el byte situado en la direccion almacenada en el par de registros HL. Por último, veamos dos ejemplos de carga de 16 bits:

LD HL,(40) copia el valor situado en la posicion 40 en el registro L, y el de la posición 41 en el registro H.
LD IY,22 carga el valor 22 en el registro de indice IY

INSTRUCCIONES ARITMETICAS DE 8 Y 16 BITS

Las más sencillas son las instrucciones de incremento y decremento, que incrementan o decrementan el dato especificado en una unidad. Sus mnemónicos respectivos son: INC y DEC.

Ejemplos:

INC A sumará uno al valor del acumulador.
DEC IX disminuirá el valor del registro IX en una unidad.

Hay dos mnemónicos, tanto para la adición, como para la sustracción: ADD, ADC, SUB y SBC. ADD y SUB son directas, el registro A se emplea para acumular el resultado de las operaciones de 8 bits, y el par de registros HL para las operaciones de 16 bits. Por ejemplo, para sumar 5 y 4, el registro A se carga con un valor: LD A,4 Después es sumado con 5: ADD A,5

El resultado se recoge en el registro A. Como dijimos antes, tanto para la adición o sustracción de 8 bits, es obligatorio utilizar el registro A. Para las operaciones aritméticas de 16 bits, el registro HL debe cargarse con uno de los valores originales, y recibe también el resultado, siendo su uso similar al del registro A en este tipo de operaciones (la excepción que confirma la regla es que el registro de índice puede emplearse como acumulador para una categoría de adicion de 16 bits).

Ejemplo: ADD HL,BC suma el contenido de BC al valor del par de registros HL. El resultado se almacena en HL.

Los mnemónicos ADC y SBC significan suma y resta con exceso. Este exceso se va a representar mediante un bit o bandera del registro de banderas. La bandera se situará a un nivel 1, si la suma produce un resultado demasiado grande para almacenarse en el acumulador, o si la resta ha requerido "llevarse una". Al ejecutar la operacion ADC, el dato especificado, más el exceso, se suman al del acumulador.

Por ejemplo ADC A,4 hará, si la bandera de exceso está a nivel 1, que se sume 5 al valor del registro A.

El juego de instrucciones incluye dos operaciones que alteran directamente el bit de exceso:

1. CCF: complementa el valor de la bandera
2. SCF: el valor de la bandera se fija en el nivel 1

No existe operación SUB para datos de 16 bits, por lo tanto no es necesario especificar el registro A, al utilizar el mnemónico SUB de 8 bits.

Ejemplo: SUB 4 disminuye el acumulador en 4 unidades

OPERACIONES LOGICAS DE 8 BITS

Los seis tipos precisan que uno de los valores sea situado en el acumulador antes de ejecutar la operación. El resultado también se almacena en el acumulador.

Tres de estas operaciones fueron presentadas con anterioridad:

1. AND
2. OR
3. XOR

Las tres restantes son:

1. CP: comparación
2. CPL: complemento de 1
3. NEG: complemento de 2

CPL invierte cada bit del acumulador. NEG también invierte cada dígito, pero incrementa el resultado para crear el complemento de 2 del valor original.

CP es anormal, en el sentido de que los datos dados se sustraen del almacenado en el acumulador. Sin embargo, no se produce resultado, por el contrario, el valor del acumulador permanece invariable, alterándose varias banderas del registro F:

1. El bit ZERO se sitúa a nivel 1, si el valor en el acumulador es igual al dato especificado. Por ejemplo CP 4 pondrá a nivel 1 la bandera Z (zero), si el acumulador contiene 4, en caso contrario, la bandera se pondrá a nivel 0.

2. El bit de signo (S) se sitúa a nivel 1, si el dato de comparación es mayor que el contenido en el acumulador. Además, la bandera N se pone a nivel 1 y las banderas H, P/V y CY son alteradas.

Ejemplos:

AND 128 utilizará el operador lógico AND siendo operandos el valor

el acumulador y 10000000.

CPL invierte cada dígito del acumulador.

OPERACIONES DE ROTACION Y DESPLAZAMIENTO

Si un valor binario se desplaza una columna a la izquierda, su valor se duplica. Este desplazamiento o rotación de un byte, es la única forma de multiplicación o división admitida por el procesador.

El Z-80 tiene tres operaciones de desplazamiento y ocho de rotación, que operan únicamente sobre datos de 8 bits. Cuando un byte es desplazado en cada uno de sus bits en una dirección, un bit extremo desaparece y otro queda vacante. El bit de exceso se utiliza para reemplazar y/o almacenar los bits desplazados o vacantes, en función de la instruccion empleada. Las ocho operaciones de rotación se clasifican en: cuatro instrucciones, que solamente rotan el byte en el acumulador, y otras cuatro, que realizan la misma operacion en cualquier registro o posicion de memoria:

1. RL y RLA: realizan idéntica operación, salvo que RLA es específica del acumulador. Cada bit se desplaza una posición hacia la izquierda. El bit menos significativo (l.s.b.) se rellena con el bit de exceso, y el bit más significativo (m.s.b.) se sitúa en el bit de exceso.

Ejemplos: RL B
RL (L)
RLA

2. RR y RRA: son iguales que RL y RLA, excepto que la rotación se produce hacia la derecha.

Ejemplos: RR H
RR (HL)
RRA

3. RLC y RLCA: aquí el bit más significativo (m.s.b.) se sitúa en el bit de exceso y en el bit menos significativo (l.s.b.).

Ejemplos: RLC (HL)
RLC A
RLCA

Tanto RLC A como RLCA realizan la misma operación. Sin embargo, al igual que sucede con otras muchas operaciones, el resultado altera varios bits del registro de bandera. RLCA y RLC A no alteran las mismas banderas.

4. RRC y RRCA: son iguales que RLC y RLCA, salvo que la rotacion se produce hacia la derecha.

Las tres operaciones de desplazamiento son:

1 SLA: desplazamiento aritmético a la izquierda.

Cada bit se desplaza a la izquierda, situándose el bit más significativo en el bit de exceso, y el menos significativo, a nivel 0

Ejemplos: SLA A
SLA (HL)

2 SRL: desplazamiento lógico a la derecha.

Cada bit se desplaza a la derecha, situando el bit menos significativo en el bit de exceso, y el bit más significativo, a nivel 0.

Ejemplo: SRL D

3. SRA: desplazamiento aritmético a la derecha.

Este es el verdadero desplazamiento aritmético, puesto que el bit de signo más significativo no se altera. El byte se desplaza una posición a la izquierda, situando el bit menos significativo en el bit de exceso. El bit mas significativo queda invariable.

Ejemplos: SRA A
SRA (HL)

Además, existen dos instrucciones para decimal codificado en binario (B.C.D.) que son RLD y RRD.

OPERACIONES DE MANEJO DE BITS

Existen dos operaciones denominadas SET y RESET que sitúan a nivel 1 ó 0, respectivamente, un bit específico de una posición determinada.

Ejemplos: SET 4,A sitúa a nivel 1 el bit 4 del acumulador.
RES 6,(HL) situa a 0 el bit 6 de la posición almacenada en el registro HL.

La instrucción BIT se utiliza para comprobar un bit específico de un registro o posición. La bandera Z se sitúa al valor complementario del bit probado, que queda invariable.

OPERACIONES DE BIFURCACION Y SUBRUTINAS

Seis de los bits del registro de bandera se utilizan como tales, los restantes dos bits, están vacantes.

S	Z	H	P	N	CY
signo	cero	medio exceso	paridad/ sobrexceso	suma/resta	exceso

Figura 4.7 Registro de banderas

Muchas de las operaciones ejecutadas por el Z-80, alteran ciertas banderas en función del resultado.

La bandera de signo (S) es una copia del bit más significativo de un resultado. Se sitúa a un nivel 1, al producirse un valor negativo (complemento de 2). La bandera de 0 (Z) solamente adquiere el nivel 1 cuando un resultado es 0. En el apéndice D, figuran aquellas operaciones que alteran la bandera Z. En dichas operaciones, un resultado distinto de cero anulará la bandera.

Las banderas de medio exceso y de suma/resta, se utilizan en operaciones de decimal codificado en binario (B.C.D.) y son de escaso interes para programadores no especializados.

La bandera de paridad/sobrexceso es anormal, en el sentido de ser alterada por ciertas instrucciones según un determinado criterio, mientras que las restantes instrucciones la modifican según criterios diferentes.

Si la bandera se renueva de acuerdo con la paridad de un resultado, se pone a 0 si el número de unos contenidos en el resultado es impar, en caso contrario, adquiere el nivel 1. Como bandera de sobrexceso alcanza el nivel 1 si una operación de suma o resta provoca que el bit 7 sea cambiado erroneamente.

En el sistema binario con signo, todos los valores negativos presentan su bit más significativo con valor 1. Un resultado fuera del rango puede producir que este bit sea invertido incorrectamente. La bandera de sobrexceso permite detectar dicha situación.

Como ya mencionamos, el registro de bandera se emplea por las instrucciones de salto condicional. Si una bandera, en particular, está a nivel 1 ó a nivel 0, se produce el salto, en caso contrario, la ejecución del programa prosigue en la siguiente instrucción.

Las instrucciones de salto fundamentales son: JR (salto relativo) y JP (salto). Ambas pueden ser condicionales o forzadas. La instrucción JR se acompaña de un byte de desplazamiento, que contiene el número de bytes a saltar desde la dirección almacenada en el contador de programa. El desplazamiento puede alcanzar hasta 127 posiciones hacia delante, ó 128 hacia detrás. En el caso de un desplazamiento negativo, se usa la forma de complemento de dos

Ejemplos: JP 40000 fuerza un salto a la posición 40000, para ello carga dicha dirección en el contador de programa. JR -120 actualiza el contador de programa para que apunte a una posición situada 120 bytes más atrás que la siguiente instrucción que hubiera tenido que ejecutar

Los códigos de condición que pueden emplearse con ambos tipos de instrucciones son:

1. Z y NZ: JP Z,40000 forzará una bifurcación si la bandera Z vale 1. JR NZ,20 realizará el salto relativo si la bandera Z vale 0.

2. C y NC: la bifurcacion se realiza si la bandera de exceso vale 1 ó 0. respectivamente. La instrucción JR utiliza solamente las condiciones Z, NZ, C y NC.

3. PO y PE: l salto se realiza si la bandera de paridad/sobrexceso, vale 1 ó 0, respectivamente. Esta condición puede utilizarse, solamente, con la instrucción JP.

4. P y M: utilizables, asimismo, solamente con la instrucción JP, forzando la bifurcación si la bandera de signo (S) vale 0 ó 1, respectivamente

Ejemplos: JP P,22000 realiza un salto a dicha posición, si la bandera de signo vale 0. JP PO,12000 ejecuta el salto si la bandera de sobrexceso vale 1

Una instrucción final de bifurcación es DJNZ, que decrementa el registro B y realiza el salto si dicho registro no es 0 Por ejemplo: DJNZ -20

La bandera Z puede colocarse a nivel 1 ó 0, mediante la instrucción BIT. Esta instrucción comprueba un bit específico de un registro o posición, y si está a 0, la bandera Z se pone a 1, y a 0 en el caso contrario. Por ejemplo: BIT 4,A situará la bandera Z al valor 1 si el bit 4 del acumulador es 0.

Las rutinas se utilizan mediante las instrucciones CALL y RET, ambas pueden operar condicionalmente mediante las condiciones descritas para la instrucción JP.

Ejemplos: CALL 20000 realiza un salto a la subrutina situada en 20000 hasta encontrar una instrucción RET. La dirección de retorno se almacena en la pila. Una llamada condicional puede realizarse en la forma: CALL NZ.12000 y un retorno condicional con: RET Z.

Una variante de la instrucción CALL es RST x donde x es múltiplo de 8, comprendido entre 0 y 56, ambos inclusive. Esta instruccion realiza una llamada a la subrutina situada en la posición x.

Por ejemplo: RST 24 realiza una llamada a la posición 24.

OPERACIONES DE TRANSFERENCIA DE BLOQUES Y DE BUSQUEDA

Hay cuatro operaciones de transferencia, y otras cuatro de búsqueda. Las instrucciones de transferencia permiten trasladar bloques de datos sin efectos sobre el acumulador:

1. LDIR: la primera posicion del bloque de datos a trasladar debe situarse en el par de registros HL, y el numero de bytes del bloque en BC. Asímismo, la direccion de destino debe colocarse en DE, todo ello antes de utilizar la instrucción. Conforme cada byte es transferido, BC se decrementa y HL y DE se incrementan. Cuando BC se iguala a 0, la transferencia ha terminado.

2 LDDR: igual en lo esencial a LDIR, excepto que las direcciones situadas en HL y DE disminuyen después de cada transferencia.

3. LDI: igual que LDIR, salvo que solamente se transfiere un byte.

4. LDD: igual que LDDR, salvo que solamente se transfiere un byte.

Ejemplo: para copiar un bloque de 20 bytes desde 12000 a 14000, haremos:

```
LD HL,12000
LD BC,20
LD DE,14000
LDIR
```

Esto mismo puede realizarse utilizando la instrucción LDDR, cargando HL con 12019 y DE con 14019.

Las instrucciones de búsqueda comprueban un bloque de datos, hasta encontrar un elemento similar al valor del acumulador:

1. CPIR: el registro BC se carga con la longitud del bloque, y HL con la direccion del primer byte del bloque. El valor a comparar se situa en el acumulador. La búsqueda finaliza cuando BC se hace 0 ó al encontrar el valor semejante. La bandera Z indica el proceso seguido:

Si vale 1: se ha encontrado un dato igual al acumulador.
Si vale 0: BC ha alcanzado el valor 0.

2. CPDR: semejante a CPIR, con la salvedad de que la búsqueda se realiza descendiendo a través de la memoria desde la dirección HL, en vez de ascender.

3. CPI: similar a CPIR, pero comprobando un único byte.

4. CPD: igual que CPDR, pero comprobando un único byte.

En todas las operaciones de movimiento y traslado de bloques, la bandera P/V indica que la cuenta ha llegado a 0:

Si BC vale 0: la bandera P/V se anula.
Si BC es distinto de 0: la bandera P/R adquiere el valor 1.

Ejemplo: para buscar un byte de valor 4 en un bloque de 120 bytes, a partir de la posición 42000, haremos:

```
LD BC,120
LD HL,42000
LD A,4
CPIR
```

Si al finalizar esta operación, la bandera 0 vale 1, el registro HL contiene la dirección del byte buscado.

INSTRUCCIONES DE CONTROL DE LA CPU Y DE E/S

El grupo de instrucciones de control de la CPU puede dividirse en cuatro secciones:

1. Operaciones de intercambio.

2. Las instrucciones NOP y HALT
3. Operaciones de pila.
4. Operaciones de interrupción y de E/S.

1. OPERACIONES DE INTERCAMBIO.

Hay dos instrucciones de intercambio. EXX intercambia el contenido de los tres pares de registros BC, DE y HL, con sus equivalentes alternativos BC', DE' y HL'. La otra operación, EX, se emplea para intercambiar el valor almacenado en dos registros de 16 bits. Presenta cuatro variantes:

1. EX AF,AF' intercambia los registros de bandera y A con F' y A'.

2. EX DE,HL intercambia el valor del par de registros DE con el del par HL.

3. EX (SP),IY ó EX (SP),IX intercambian los registros de índice con los dos bytes superiores de la pila.

4. EX (SP),HL similar al caso 3, excepto que el par de registros HL ocupa el lugar del registro de índice.

2. INSTRUCCIONES NOP y HALT.

La instrucción NOP (NO Operación) no hace absolutamente nada durante cuatro intervalos de reloj. Se emplea, frecuentemente, para reemplazar código redundante o para rellenar una zona que puede utilizarse para una futura ampliación del programa.

La operación HALT provoca que el microprocesador ejecute, repetidamente, la instrucción NOP, hasta que se produzca una interrupción o reposición (RESET). Las instrucciones NOP se realizan para permitir que el Z-80 "refresque" su RAM dinámica. El registro de refresco se incrementa automáticamente después de recoger cada instrucción de la memoria. El valor almacenado corresponde al byte de orden más bajo de la zona de memoria que precisa ser refrescada.

3. OPERACIONES DE PILA.

Todas las instrucciones de pila operan sobre datos de 16 bits. El contenido de cualquier par de registros puede ser colocado en la pila, mediante la instrucción PUSH. El valor de los dos bytes superiores de la pila puede, a su vez, ser cargado en cualquier par de registros con la instrucción POP. La ejecución de PUSH no altera los contenidos de los registros.

Al completarse la operación, el puntero de pila contiene la dirección del byte superior de la pila. Por lo tanto, si el puntero de pila contiene X, el byte de mayor orden del último registro alojado en

la pila, se situó en la posición X-1, y el byte de menor orden en la posicion X.

Ejemplos:

PUSH BC, POP BC, PUSH IY, POP IY

El registro del puntero de pila puede alterarse directamente utilizando INC, DEC ó LD.

Ejemplos:

INC SP, INC SP aumentará el puntero dos posiciones Como la pila crece hacia abajo en la memoria, los 16 bits iniciales de la pila se pierden.

4 OPERACIONES DE INTERRUPCION Y DE ENTRADA/SALIDA.

Operaciones de interrupción: un dispositivo externo puede interrumpir, temporalmente, las actividades usuales de ejecución del procesador. Esto se consigue utilizando alguna de las cuatro líneas de control del Z-80:

1. BUSRQ
2. RESET
3 NMI
4. INT

Las líneas 1-3 se usan muy poco por el programador. BUSRQ, presentado anteriormente en el capítulo, permite que otro procesador utilice las direcciones del sistema y los buses de datos. RESET se emplea para inicializar el sistema al conectar la alimentación. Los registros I y R se anulan antes de comenzar la ejecución del programa desde la posicion 0.

La interrupción no enmascarable (NMI) fuerza al procesador a realizar una llamada a una subrutina especial situada en la posición &H66. En la configuración MSX, esta posición está asignada para uso del sistema operativo de disco, y por tanto, el NMI será raramente utilizado.

La línea INT puede comenzar una de tres secuencias. El modo se selecciona por el programador con la instrucción apropiada: IM 0, IM 1 ó IM 2 En cada caso, el contenido del registro de contador de programa, se guarda en la pila, y se ejecuta un salto a la posición especificada. Al encontrar la instrucción RETI (RETorno de Interrupción), el contador de programa se carga con la posición de retorno, y se vuelve a comenzar la rutina interrumpida.

La entrada INT se comprueba por el Z-80 después de ejecutar cada instruccion. Dicha entrada es sensible al nivel, no al impulso. Por tanto, el dispositivo que interrumpe, no debe mantener, continuamente, la linea a nivel bajo, salvo que se precise otra interrupcion.

El procesador tiene dos biestables internos, que se denominan IFF1 e IFF2 en el proceso de interrupción. Si IFF1 se pone a un nivel lógico

0, cualquier llamada a través de INT, se ignora por el procesador. Si se pone a un nivel lógico 1, se activan las interrupciones enmascarables.

El programador puede fijar los estados de IFF1 e IFF2 utilizando las instrucciones EI (activación de interrupción) o DI (desactivación de interrupción). IFF2 se emplea para almacenar el estado de IFF1 durante un NMI, momento en el cual IFF1 se pone a nivel 1 para impedir las llamadas a través de INT

La dirección de la rutina ejècutada se selecciona mediante procedimientos diferentes para cada uno de los tres modos de INT:

Interrupción modo 0: activada por la instrucción IM 0

1. IFF1 y IFF2 se ponen a nivel lógico 0. Se evitan, así, problemas de reentrada.

2. El Z-80 reconoce la interrupción durante el siguiente pulso de reloj, debido al nivel bajo de las dos líneas de control, IORQ y M1.

3. El dispositivo sitúa un código de operación de un byte en el bus de datos. Este puede ser una instrucción RST o CALL.

4. Si se dá un código RST, el contenido del registro de contador de programa se sitúa en la pila, y se realiza a la posición relevante de la página 0.

5. Si se dá un código de llamada, el dispositivo debe situar dos bytes de datos adicionales en el bus de datos. La rutina fijada por dicha dirección se ejecuta después de situar el registro de contador de programa en la pila.

6 Al encontrar una instrucción RETI, los dos bytes de datos de la parte superior de la pila, se vuelven a cargar en el contador de programa. La instrucción RETI no fija a IFF1 a un nivel lógico 1, por lo que las interrupciones deben reactivarse mediante EI.

La instrucción RET también puede emplearse para forzar un retorno a la rutina inicial. Sin embargo, algunos dispositivos son capaces de detectar que el Z-80 ha recogido una instrucción RETI de la memoria Esta posibilidad puede emplearse para eliminar la petición inicial de interrupción.

Interrupción modo 1: activada por la instrucción IM 1.

La secuencia de inicialización, ejecutada por un ordenador MSX, al ser conectada la alimentación, se realiza mediante este modo. La secuencia ejecutada después de haberse producido una petición activada, es la siguiente:

1. Los biestables IFF1 e IFF2 se ponen a 0 lógico

2. El contenido del contador de programa se colocan en la pila

3. La ejecucion del programa continúa a partir de la instrucción situada en &H38

4. Al encontrar una instrucción RETI, los dos bytes superiores de la pila, se cargan en el contador de programa, y vuelve a comenzar la ejecución de la rutina interrumpida.

Para reactivar las interrupciones enmascarables, IFF1 debe ponerse a nivel 1 lógico mediante la instrucción EI.

Interrupción modo 2: activada por la instrucción IM 2

El modo 2 es el más flexible de los tres modos de interrupción Permite que un dispositivo acceda hasta 128 rutinas separadas Después de que la interrupción es detectada por el procesador, el dispositivo sitúa un único byte en el bus de datos, que se une al byte del registro I para formar una dirección de 16 bits:

Byte del registro I / byte del dispositivo
b b b b b b b b b b b b b b

El puntero de dos bytes en esta dirección dará la posición de la rutina a ejecutar:

Dirección de 16 bits	Byte de orden alto Byte de orden bajo	Puntero de 16 bits a la rutina de servicio

La secuencia ejecutada es:

1. IFF1 toma el valor 0 lógico.

2. El Z-80 reconoce la interrupción mediante las líneas IORQ y M1.

3. El dispositivo sitúa un byte de datos en el bus de datos.

4. La dirección de ejecución del contador de programa se almacena en la pila.

5. Se ejecuta la rutina señalada por el puntero de la posición calculada

6. Al encontrar una instrucción RETI, los dos bytes superiores de la pila se asignan al registro del contador de programa, y se continúa la rutina interrumpida.

OPERACIONES DE ENTRADA/SALIDA

Este grupo tienen dos conjuntos de instrucciones para la entrada y salida de datos a un puerto de 8 bits.

1 Entrada/Salida de un único byte:

Los dos mnemónicos son IN y OUT. Cada uno tiene dos formas, la primera utiliza solamente el registro A, y la segunda, cualquiera de los siguientes: A, B, C, D, E, H ó L.

Si se emplea la variante del acumulador, el puerto debe ir entre paréntesis: IN A,(puerto). La segunda variante utiliza el registro C

para seleccionar el puerto: OUT (C),registro.

Ya hemos discutido brevemente la mecánica de las operaciones de E/S El Z-80 envía el número de puerto mediante las 8 líneas menos significativas del bus de direcciones. Previamente a situar el byte de datos en el bus de datos.

Sin embargo, hay que destacar que tanto IN A,(puerto) y OUT (puerto),A también colocan el contenido del acumulador en las líneas de direcciones A15 a A8. Asimismo, las operaciones IN registro,(C) y OUT (C),registro situan el contenido del registro B en las líneas de direcciones superiores. Se podrian utilizar, así, mas de 256 puertos, siempre que se dispusiera de los circuitos necesarios para su decodificacion.

2. Entrada/Salida de bloques:

El bloque de datos puede contener desde 1 a 256 bytes Este tipo de instrucciones se clasifica en cuatro grupos, y es similar, en su estilo, a las categorías de transferencia y búsqueda:

1. INIR Y OTIR: los datos de inicializacion precisos son:

a Número de puerto en el registro C.
b. Primera posición de datos en HL.
c. Tamaño del bloque en el registro B

El par de registro HL se incrementa después de cada transferencia Si el registro B alcanza el valor 0, la bandera Z se pone a nivel 1.

2 INDR y OTDR: similar a INIR y OTIR, salvo que HL disminuye después de cada transferencia.

3. INI y OUTI: similar a INIR y OTIR, salvo que solamente se produce una transferencia. HL se incrementa y B disminuye.

4 IND y OUTD: similar a INI y OUTI, salvo que HL disminuye después de la transferencia.

MODOS DE DIRECCIONAMIENTO

Como hemos visto anteriormente, cada mnemonico de instrucción puede especificar la posicion de los datos objetivo de una o varias formas. Aunque el Z-80 tiene diez modos de direccionamiento, no es necesario que cada tipo de instruccion disponga de una variante para cada modo El lector encontrara que la mayoría de los formatos -muchos de los cuales ya han sido presentados- son comprensibles directamente:

1. Implícito: No hace falta dirección, está implícita en la instrucción.

Ejemplo: NEG

2. Inmediato: el dato de 8 bits está dado, en vez de darse su dirección

Ejemplo: LD A,2

3 Inmediato extendido: el dato de 16 bits está dado, en vez de emplearse su direccion.

Ejemplo: LD HL,20000

4 Relativo: este modo se emplea únicamente por las instrucciones de salto relativo y DJNZ El byte siguiente a la instrucción contiene un valor con signo comprendido entre -128 y 127, ambos inclusive Este desplazamiento se suma a la dirección del contador de programa para dar la nueva dirección de ejecución.

Ejemplo: JR 20

5 Registro: el dato a utilizar está contenido en el registro especificado.

Ejemplo: LD A,B

6. Registro indirecto: un registro o par de registro de índice, contiene la dirección del dato a utilizar. El registro o par de registro se escribe entre parentesis.

Ejemplo: LD A,(DE)

7. Extendido: la dirección del dato se dá entre paréntesis.

Ejemplo: LD A,(4000)

8. Indexado: es esencialmente la misma forma de registro indirecto, pero utilizando un desplazamiento que se suma a la dirección almacenada por el registro de índice.

Ejemplo: LD A,(IY+5)

9. Página cero modificada: solamente las instrucciones RST utilizan este formato. Se ejecuta una llamada a la posición especificada, que debe ser multiplo de 8 y estar comprendida entre 0 y 56, ambos inclusive

Ejemplo: RST 8

10. Bit: el modo de bit es anormal, puesto que especifica un bit y no un byte. El modo de byte se emplea por tres tipos de instrucciones: BIT, SET y RES.

Los bits están numerados como sigue:

X	X	X	X	X	X	X	X
7	6	5	4	3	2	1	0

Ejemplo: SET 4,A

Hemos alcanzado una fase en la que los lectores pueden desear probar algunas de sus rutinas Las siguientes directrices pueden ser de gran ayuda:

1 Se debe guardar el programa en código máquina, antes de ejecutarlo. Si contiene algún error, el ordenador, probablemente, quedará "colgado"

y será necesario desconectarlo antes de volverlo a utilizar.

2. Es difícil que una rutina funcione correctamente desde el principio. Con suerte, una o dos partes actuarán según lo esperado. Paciencia, es la palabra clave.

3. Si se combina una rutina en código máquina con un programa en BASIC, es más sencillo utilizar un bucle en BASIC para colocar el código en memoria, que cargar y guardar dos programas separados.

Si se precisa traspasar valores entre ambos programas, y salvo que se utilicen las rutinas de P.F. residentes, se deben emplear las instrucciones PEEK y POKE, con preferencia a la función USR.

4. Normalmente, las rutinas en código máquina precisan más depuración, que sus equivalentes en BASIC, y es aconsejable estructurar el programa en pequeñas secciones, cada una de las cuales es llamada desde un programa núcleo.

CAPITULO 5

La Configuracion del Sistema MSX

La configuración básica del sistema MSX utiliza una CPU Z-80 (o equivalente), funcionando a una velocidad de 3.5 MHz, auxiliada por un procesador de video de Texas Instruments TMS-9929 A (TMS-9918 A en América y Japón), y un circuito de sonido de General Instruments AY-3-8910.

Asímismo, se emplea un interfaz programable para periféricos Intel 8255, que controla la exploración del teclado y el sistema de paginación de memoria. La especificación se completa con el BASIC MSX y el sistema operativo que ocupan 32 Kbytes de ROM y pór un mínimo de 8 Kbytes de RAM de usuario (aunque para el mercado europeo es muy improbable que se oferte máquinas con menos de 32 Kbytes de RAM). Aparte de esto, el VDP (procesador de video) dispone de 16 Kbytes de RAM dedicada a video, que emplea para almacenar la memoria de pantalla, el color y las definiciones de sprite. La utilización de esta "VRAM" será discutida, ampliamente, en el capítulo 6.

GESTION DE LA MEMORIA EN MSX

Para comprender la gestión de la memoria en los sistemas MSX, es fundamental el concepto de "ranura", que describiré, a continuación, brevemente.

Como el Z-80 es un procesador de 8 bits, su máximo espacio direccionable es de 64 Kbytes; sin embargo, el sistema MSX está diseñado para acceder hasta 1024 Kbytes, para lo que emplea el sistema de ranuras.

Una ranura es un conjunto de 64 Kbytes de espacio direccionable. El sistema MSX puede direccionar, en principio, 4 de éstas ranuras. Cada una de éstas 4 ranuras primarias, puede expandirse posteriormente a 4 ranuras secundarias, dando así un espacio direccionable total de 4*4*64=1024 Kbytes. Cada ranura (primaria o secundaria) está subdividida en 4 páginas de 16 Kbytes, que son las unidades básicas empleadas por el sistema MSX para crear su espacio de direcciones. Cada una de las 4 páginas distintas comienzan en las direcciones 0000H, 4000H, 8000H y C000H, independientemente de la ranura en que se encuentren. Las páginas solamente se pueden direccionar en su dirección respectiva de

página (la página 0 en 0000H, la página 1 en 4000H, y así, sucesivamente). Por lo tanto, el sistema MSX puede seleccionar cualquier página de cualquier ranura, pero siempre la página 0 se direccionará en 0000H, y, por ejemplo, la página 3 en C000H.

La selección de ranuras primarias se realiza escribiendo, según el siguiente formato, en el puerto A del 8255 PPI:

```
Bits:      7 6         5 4         3 2         1 0
           :1 0        :0 1        :1 1        :0 0        Puerto A 8255
            |           |           |           |
        página 3    página 2    página 1    página 0
        ranura 2    ranura 1    ranura 3    ranura 0
```

Cada asignación de ranura utiliza dos bits del registro, por lo tanto, cada asignación puede variar numéricamente entre 0 y 3 (00B y 11B), permitiendo así la selección de cada una de las cuatro ranuras para cualquiera de las cuatro páginas. La selección de la ranura secundaria se consigue escribiendo, en el mismo formato, la posición FFFFH de la ranura primaria. Cuando se lee este registro, el valor obtenido es el complemento del escrito anteriormente.

Al conectar la alimentación o realizar una reposición del sistema MSX (que reside en la ranura 0, que es la primera en ser leida por la CPU), éste busca el mayor espacio contiguo de memoria RAM a partir de FFFFH hacia abajo y selecciona dicha área para las páginas 3 y 2. En circunstancias normales no es necesario, por tanto, que el programador realice la selección de ranuras, excepto en el caso de un sistema de 64 Kbytes o más de RAM, en el cual la RAM se superpone a la ROM, o cuando se utilicen programas de cartuchos diseñados para ocupar las ranuras distintas de cero.

ACCESO A LOS CIRCUITOS DE SONIDO, VIDEO Y PERIFERICOS

El Z-80, además de direccionar 64 Kbytes de memoria, también puede direccionar 256 puertos de E/S. Con ellos, se controla el VDP, etc. (incluyendo la conexión RS232 y otras expansiones del sistema). Sin embargo, en orden a permitir modificaciones de los circuitos, se han previsto llamadas al BIOS (sistema de entradas/salidas de BASIC) en la ROM del sistema MSX. Se permite, así, el acceso a los circuitos del sistema MSX, de forma tal, que si en una unidad MSX, uno de los circuitos ocupa una posición diferente, puede ser compensada mediante modificaciones del programa monitor en fábrica. La diferencia es imperceptible para el programador, y asegura completa compatibilidad entre los programas. La única excepción a esta regla es el VDP, que será discutido en profundidad más adelante.

INTRODUCCION GENERAL AL VDP

El procesador de pantalla de video de Texas Instruments TMS-9929 A es un microprocesador por derecho propio. Es responsable de todo el

control de la pantalla, y accede a 16 Kbytes de memoria RAM exclusiva. Esta RAM contiene todas las definiciones de caracteres, los datos de pantalla, los datos de color, las definiciones de sprite y los atributos de sprite. Se libera, así, la RAM del sistema para usarse en programación (la penalización pagada consiste en una reducción en la velocidad de acceso en la RAM de video). En circunstancias normales, esta reducción no es crítica. En situaciones especiales, pueden emplearse algunos trucos que se discutirán en el capítulo 6.

El VDP tiene cuatro modos de pantalla: Gráficos I, Gráficos II, Multicolor y Modo de Texto. El modo de texto dispone de 24 filas de 40 caracteres en dos colores, y fue diseñado para maximizar la capacidad de visualización de textos de las pantallas de televisión. El modo multicolor permite mostrar una pantalla de 64*48 puntos de color, utilizando sin limitación 15 colores más el transparente. El modo de gráficos I genera una pantalla de 256*192 puntos con capacidad para mostrar patrones gráficos en 15 colores más el transparente. El modo de gráficos II es una potenciación del modo de gráficos I, permitiendo la generación de patrones y colores más complejos.

La pantalla de video está compuesta por 34 planos apilados verticalmente. El plano inferior es el de fondo, sobre él se encuentra el plano de patrón (que contiene los patrones de los modos gráficos I y II), y sobre éstos existen 32 planos de sprite.

ADemás del manejo de la pantalla de video, el VDP también refresca la RAM de usuario, y realiza una llamada de interrupción cada 20 microsegundos. Para más detalles sobre el VDP y su programación, vease el capítulo 6.

EL CIRCUITO DE SONIDO GENERAL INSTRUMENTS AY-3-8910

El AY-3-8910 de G.I. es un circuito de sonido con tres voces y ocho octavas, que incorpora envolvente de volumen y la posibilidad de mezclar una señal de ruido con una o todas las voces. El circuito también incorpora dos puertos de E/S empleados por el sistema para el control de mandos de juego, raquetas y tablero digitalizador.

Para una descripción de la programación de este circuito, veáse el capítulo 7.

EL INTERFAZ PROGRAMABLE DE PERIFERICOS INTEL 8255

El PPI 8255 es un circüito de E/S muy potente, de usos generales, empleado en el sistema MSX para varias funciones. Es responsable del control del teclado (incluyendo sus extensiones), la lámpara de mayúsculas, varias funciones de E/S a cassette, el sistema de gestión de memoria y, además, dispone de un puerto de sonido de 1 bit.

Una descripción más completa de este circuitos y de sus usos múltiplos, se encuentra en el capítulo 8.

CONTROL DE INTERRUPCIONES Y "GANCHOS EN RAM"

Muchas de las rutinas contenidas en ROM, se direccionan a través de la RAM, mediante el empleo de "ganchos". En el sistema MSX, un "gancho" consiste en 5 bytes de RAM inicializados para contener una instrucción RET del Z-80 (C9H). El direccionamiento se realiza por la rutina en ROM correspondiente, mediante una instrucción CALL del Z-80 al "gancho". Por tanto, es sencillo redirigir la llamada (CALL) insertando una instrucción JP nnnn de Z-80, en los primeros 3 bytes del gancho.

El sistema MSX funciona en el modo de interrrupción 1, y dispone de dos ganchos para el manejo de las peticiones de interrupción (IRQ).

El primero de ellos está situado en FD9AH, y se emplea para el control de los IRQs originados en dispositivos diferentes del temporizador del VDP. En el sistema MSX básico, este gancho casi no se utiliza, pero está disponible para una expansión posterior. El segundo gancho se encuentra en FD9FH y se utiliza para el contol de las interrupciones genradas en el retorno de barrido de pantalla por el VDP. Esta es la única interrupción disponible en el sistema básico. La interrupción del VDP también controla la exploración del teclado y varias funciones del sistema operativo. Es esencial que en el retorno de cualquier rutina de usuario se produzca a la rutina de llamada del sistema operativo, en orden a manejar la interrupción limpiamente.

El sistema operativo sitúa todos los registros, incluyendo el juego alternativo, en la pila, antes de llamar al gancho situado en FD9FH. Por lo tanto, pueden utilizarse todos los registros libremente. Sin embargo, el sistema operativo también recoge el contenido del registro de estado del VDP y lo almacena en el acumulador, por lo tanto es esencial que cualquier rutina de usuario no modifique el par AF. Al volver del gancho, el sistema operativo almacena el valor del acumulador en la posición F3E7H, siendo preferible leer esta posición en vez del registro de estado del VDP, al comprobar cualquiera de los bits de estado del VDP en una rutina de usuario que no sea dirigida mediante interrupciones.

Además de estos ganchos, el sistema también dispone de otro en la posicion FDD6H, para el manejo de las interrupciones no enmascarables (NMI). Su uso es restringido, puesto que el sistema básico no dispone de NMIs y en el sistema de disco el vector de entrada de NMI situado en 66H esta ocupado por los datos del bloque de control de archivos del sistema operativo de disco.

En aplicaciones de usuario, el manejo mediante de NMIs puede ser necesario, por lo tanto, este gancho permite una mayor flexibilidad.

El siguiente programa genera un reloj de tiempo real dirigido, mediante enterrupciones, como ejemplo de utilización de una IRQ por el usuario.

PROGRAMA EJEMPLO: RELOJ DE TIEMPO REAL

```
  10 WRTVDP:   EQU  #0047            ;TABLA DE LLAMADAS A
BIOS
  20 RDVRM:    EQU  #004A
  30 WRTVRM:   EQU  #004D
  40 FILVRM:   EQU  #0056
  50 LDIRVM:   EQU  #005C
  60 LDIRMV:   EQU  #0059
  70 CHGET:    EQU  #009F
  80 CHPUT:    EQU  #00A2
  90 GTSTCK:   EQU  #00D5
 100 GTTRIG:   EQU  #00D8
 110 RDVDP:    EQU  #013E
 120 SNSMAT:   EQU  #0141            ;FIN DE TABLA DE BIOS
 130 VDPRGS:   EQU  #F3DF            ;ZONA DE ALMACENAMIEN
TO DE REGISTROS DEL VDP
 140 BASNOS:   EQU  #F3B3            ;POSICION DE COMIENZO
 DE LA TABLA DEL VDP
 150 INTHOK:   EQU  #FD9F
 160           ORG  #E000
 170 ;
 180 ;
 190 START:    LD   HL,MESS          ;SE FIJA HL EN EL PUN
TO DE ESPERA DE ENTRADA
 200 MPLP:     LD   A,(HL)
 210           CP   "$"
 220           JR   Z,GETIN
 230           CALL CHPUT            ;IMPRIME ESPERA DE DA
TOS
 240           INC  HL
 250           JR   MPLP
 260 GETIN:    LD   HL,HHMMSS
 270           LD   B,3
 280 GTINOL:   CALL GETNUM           ;ENTRADA DE 2 NIBBLES
 NUMERICOS
 290           LD   (HL),A           ;SE GUARDAN EN HHMMSS
 300 GTINIL:   CALL CHGET            ;ENTRADA DE UN CARACT
ER
 310           CP   ":"              ;ES EL SEPARADOR?
 320           JR   NZ,GTINIL        ;SI NO PRUEBA DE NUEV
O
 330           INC  HL               ;APUNTA AL SIGUIENTE
DE HHMMSS
 340           DJNZ GTINOL           ;HAZLO TRES VECES
 350           PUSH HL
 360           LD   HL,CLOCK
 370           LD   A,L
 380           LD   (INTHOK+1),A     ;SALTA A DIRECCION
 390           LD   A,H
```

```
 400                 LD    (INTHOK+2),A
 410                 POP   HL
 420                 LD    A,#00C3              ;SALTO INCONDICIONAL
Z-80
 430                 LD    (INTHOK),A
 440                 RET                        ;A BASIC DESPUES DE R
EDIRIGIR EL GANCHO
 450 ;
 460 GETNUM:         CALL  CHGET                ;CONSIGUE CARACTER
 470                 CP    "0"
 480                 JR    C,GETNUM
 490                 CP    ":"                  ;ES NUMERICO?
 500                 CALL  CHPUT                ;IMPRIMELO
 510                 SUB   "0"                  ;CONVIERTE A NUMERICO
 520                 SLA   A
 530                 SLA   A
 540                 SLA   A                    ;MULTIPLICA POR 16
 550                 SLA   A
 560                 LD    B,A                  ;GUARDALO EN B
 570 GTNUM1:         CALL  CHGET                ;CONSIGUE CARACTER
 580                 CP    "0"
 590                 JR    C,GTNUM1
 600                 CP    ":"                  ;ES NUMERICO?
 610                 JR    NC,GTNUM1            ;SI NO PRUEBA DE NUEV
O
 620                 CALL  CHPUT                ;IMPRIMELO
 630                 SUB   "0"                  ;CONVIERTE A NUMERICO
 640                 ADD   A,B                  ;SE SUMA A B
 650                 RET                        ;VUELVE CON UN NUMERO
EN A
 660 MESS:           DEFM "INTRODUCE LA HORA EN HH:MM:SS:$"
 670 HHMMSS:         DEFB 0,0,0
 680 CTR:            DEFB 0
 690 CLOCK:          PUSH AF                    ;GUARDA EL ESTADO DEL
VDP
 700                 LD    A,(CTR)
 710                 DEC   A
 720                 LD    (CTR),A              ;INTERVALO DE RELOJ C
ADA 50...
 730                 JP    P,OUT                ;INTERRUPCIONES
 740                 LD    A,49
 750                 LD    (CTR),A              ;REPOSICION DE CONTAD
OR
 760                 LD    IX,HHMMSS
 770                 LD    A,(IX+2)
 780                 INC   A                    ;INCREMENTA SEGUNDOS
 790                 DAA
 800                 LD    (IX+2),A
 810                 CP    #0060                ;SI ES NECESARIO
```



```
 820                 JR    NZ,CLKPNT
 830                 XOR   A                    ;REPOSICION DE SEGUND
OS
 840                 LD    (IX+2),A
 850                 LD    A,(IX+1)
 860                 INC   A                    ;INCREMENTA MINUTOS
 870                 DAA
 880                 LD    (IX+1),A
 890                 CP    #0060                ; SI ES NECESARIO
 900                 JR    NZ,CLKPNT
 910                 XOR   A                    ;REPOSICION DE MINUTO
S
 920                 LD    (IX+1),A
 930                 LD    A,(IX+0)
 940                 INC   A                    ;INCREMENTA HORAS
 950                 DAA
 960                 LD    (IX+0),A
 970                 CP    #0024
 980                 JR    NZ,CLKPNT
 990                 XOR   A
1000                 LD    (IX+0),A
1010 CLKPNT:         LD    HL,VDPRGS            ;RUTINA PARA IMPRIMIR
...
1020                 LD    A,(HL)               ;EL RELOJ RENOVADO
1030                 LD    DE,10                ;LEE LA COPIA DE LA R
AM DEL SISTEMA...
1040                 AND   2                    ;DE LOS REGISTROS DE
ESCRITURA...
1050                 JR    NZ,GNAMTB            ;DEL VDP
1060                 INC   HL                   ;PARA ENCONTRAR EL MO
DO DE PANTALLA...
1070                 LD    A,(HL)               ;Y CALCULAR EL DESPLA
ZAMIENTO...
1080                 AND   8                    ;DESDE EL COMIENZO DE
 BASE(N)...
1090                 JP    NZ,OUT               ;QUE CONTIENE LA TABL
A DE VARIABLES...
1100                 LD    A,(HL)               ;PARA ENCONTRAR LA DI
RECCION DE BASE...
1110                 AND   16                   ;DE LA TABLA DE NOMBR
ES
1120                 LD    DE,0
1130                 JR    NZ,GNAMTB
1140                 LD    DE,5
1150 GNAMTB:         LD    IX,BASNOS
1160                 ADD   IX,DE
1170                 LD    L,(IX+0)             ;CONSIGUE LA DIRECCIO
N DE LA TABLA DE NOMBRES
1180                 LD    H,(IX+1)
```

```
 1190           LD    DE,24           ;LE SUMA 24 (DESPLAZA
MIENTO)
 1200           ADD   HL,DE
 1210           LD    IX,HHMMSS       ;CONSIGUE LA DIRECCIO
N DE LA HORA
 1220           LD    A,(IX+0)        ;CONSIGUE EL PRIMER V
ALOR(HORA)
 1230           CALL  NPNT            ;LO IMPRIME
 1240           LD    A,":"           ; CONSIGUE EL SIGNO:
 1250           CALL  WRTVRM          ;LO IMPRIME
 1260           INC   HL              ;INCREMENTA LA POSICI
ON DE IMPRESION
 1270           LD    A,(IX+1)        ;CALCULA MINUTOS
 1280           CALL  NPNT
 1290           LD    A,":"
 1300           CALL  WRTVRM
 1310           INC   HL
 1320           LD    A,(IX+2)        ;CALCULA SEGUNDOS
 1330           CALL  NPNT
 1340 OUT:      POP   AF              ;RESTAURA EL ESTADO D
EL VDP...
 1350           RET                   ;VUELVE A LA RUTINA D
EL S.O.
 1360 NPNT:     PUSH  AF              ;GUARDA EL VALOR NUME
RICO
 1370           SRL   A               ;CONSIGUE EL NIBBLE S
UPERIOR
 1380           SRL   A
 1390           SRL   A
 1400           SRL   A
 1410           ADD   A,#0030         ;LO CONVIERTE A ASCII
 1420           CALL  WRTVRM          ;LO IMPRIME
 1430           INC   HL              ;INCREMENTA LA POSICI
ON DE IMPRESION
 1440           POP   AF              ;RESTAURA EL VALOR NU
MERICO
 1450           AND   15              ;CONSIGUE EL NIBBLE I
NFERIOR
 1460           ADD   A,#0030         ;LO CONVIERTE A ASCII
 1470           CALL  WRTVRM          ;LO IMPRIME
 1480           INC   HL              ;INCREMENTA LA POSICI
ON DE IMPRESION
 1490           RET
 1500 ;
 1510           END
```


UTILIZACION DE LA RAM EN EL SISTEMA MSX

En el sistema MSX las posiciones desde F380H hasta FFFFH se emplean por BASIC y el sistema operativo, para funciones de control y mantenimiento del sistema. Algunas de estas posiciones, de mayor utilidad para el programador en ensamblador, se describen a continuación.

Las rutinas de llamada y lectura/escitura entre ranuras se encuentran al comienzo de la RAM del sistema.

En F380 encontramos una rutina para realizar la lectura de cualquier ranura primaria, que acepta el valor para la selección de ranura (en el formato requerido para el 8255), en el acumulador, el antiguo valor de ranura en el registro D, y la dirección en el par de registros HL. La rutina proporciona el valor leido en el registro E, preservando los restantes registros.

La rutina complementaria a ésta se encuentra en F385H, empleando los mismos valores iniciales, pero realizando una operación de escritura con el dato contenido en E.

Por último, la rutina situada en F38CH, realiza llamadas entre ranuras. Esta rutina precisa que se sitúe el estado de la rutina anterior en la pila, el estado de la nueva rutina en el acumulador, cualesquiera valores a trasmitir, en el par alternativo AF y la posición a llamar en IX. Los valores de retorno se encontrarán en el par alternativo AF.

El sistema almacena las direcciones de llamada de las funciones USR de BASIC en las posiciones F39AH a F3ADH, en el orden USR0 a USR9, con dos bytes para cada dirección. Las posiciones F3B3H a F3D9H contienen los valores a los que ha accedido la seudovariable BASE (N) de BASIC, también con dos bytes por valor.

El indicador de eco de teclado se almacena en F3DBH. Si se escribe un cero en esta posición, se desactiva el eco de teclado, mientras que si se escribe un valor distinto de cero, se activa.

La posición (X,Y) del cursor de pantalla se almacena en las posiciones F3DCH (Y) y F3DDH (X).

Las ocho posiciones desde F3DFH en adelante, se emplean para almacenar el color de primer plano, de fondo, y de borde, definidas por la instrucción COLOR de BASIC.

El sistema almacena la dirección de la mayor posición encontrada en RAM en las direcciones F672H y F673H. Estas posiciones pueden alterarse para reservar zonas de RAM del BASIC y sistema operativo, protegiendo cualquier programa de máquina realizado por el usuario, almacenado en dichas zonas. Los dos bytes siguientes a estas posiciones definen la dirección más alta empleada por la pila.

Las 26 posiciones que comienzan en F6CAH se utilizan para almacenar los tipos de variable asumidos por omisión, para las variables que empiezan por A-Z. Esta tabla se modifica por DEFINT, DEFSTR, DEFSNG, etc. y se lee al encontrar una variable que no vaya acompañada por la

declaración de tipo de variable (por ejemplo $, %, !). Los tipos de variable son:

Entera	: valor en la tabla 2
Cadena	: valor en la tabla 3
Precisión sencilla	: valor en la tabla 4
Precisión doble	: valor en la tabla 8

Los elementos de la tabla por omisión, corresponden a precisión doble.

Por último, desde FD9AH hasta FFCAH, se encuentra el bloque de RAM conteniendo los ganchos, cada uno de los cuales consta de 5 bytes, que inicialmente contienen instrucciones RET del Z-80 (C9H). La mayoría de ellos están previstos para una expansión futura del sistema, y son de poca utilidad. Aquellos de uso inmediato, serán explicados conforme vayan apareciendo.

UTILIZACION DE SUBRUTINAS EN CODIGO MAQUINA DESDE BASIC

La clave para enlazar con éxito las rutinas de código máquina con el lenguaje BASIC consta de dos partes: primero hay que proteger la rutina para impedir que BASIC escriba sobre ella, y segundo, hay que pasar parámetros y leerlos de la rutina.

El método más simple para asignar espacio para las rutinas de máquina, consiste en emplear la sentencia CLEAR n1, n2 desde BASIC. El primer parámetro fija la cantidad de espacio para almacenamiento de cadenas, y el segundo, determina la dirección más alta de memoria que puede utilizarse por BASIC. Por lo tanto, para liberar los 16 kbytes superiores de la RAM, y asi emplearlos con rutinas de máquina y datos, y además para obtener 200 bytes de espacio para cadenas, haremos CLEAR 200,&HBFFF En circunstancias normales el área de trabajo del sistema ocupará las posiciones F380H en adelante.

Para ejecutar las rutinas en código máquina, desde BASIC, se emplea la función USRn. La sintaxis de esta sentencia es:

Var = USRn (Var/Const) ó PRINT USRn (Var/const)

donde n es igual al número de rutina de usuario definido por DEFUSR y Var ó Var/const puede ser de cualquier tipo.

Si queremos pasar una cadena a una rutina que se pretende que devuelva un valor entero, podemos hacer:

A% = USR1 ("abcde")

La función USR pasa los parámetros de la siguiente forma:

Entero: la posición F633H contendrá 2, y el valor se almacena en F7F8H y F7F9H (primero el byte más bajo).

Cadena: la posición F663H contendrá 3, y F7F8H y F7F9H contendrán la dirección del descriptor de cadena. Un descriptor de cadena consiste en 3 bytes; la longitud de la cadena seguida por su dirección.

Precisión sencilla: la posición F663H contendrá 4, y el valor se almacena desde F7F6H hasta F7F9H.

Precisión doble: la posición F663H contendrá 8, y el valor se almacena desde F7F6H hasta F7FDH.

Los parámetros retornan a BASIC en forma similar. El siguiente programa explica el proceso aceptando una cadena numérica en base 3, y retorna el valor decimal como entero.

```
  10 VARTYP:   EQU  #F663
  20 VARPTR:   EQU  #F7F8
  30           ORG  #E000
  40 START:    LD   A,(VARTYP)
  50           CP   3                 ; ES VARIABLE DE CADE
NA?
  60           RET  NZ                ;NO, RETORNA
  70           LD   HL,(VARPTR)       ;COGE DIRECCION DESCR
IPTOR
  80           LD   B,(HL)            ;LONGITUD DE CADENA E
N B
  90           INC  HL
 100           LD   E,(HL)            ;DIRECCION DE CADENA
EN DE
 110           INC  HL
 120           LD   D,(HL)
 130           LD   HL,0              ;BORRA HL
 140 ICLP:     PUSH DE
 150           PUSH HL
 160           ADD  HL,HL
 170           POP  DE
 180           ADD  HL,DE             ;VALOR BIN=VALOR BIN*
3
 190           POP  DE                ;COGE PUNTERO CADENA
 200           EX   DE,HL             ;LO PONE EN HL, VALOR
 BIN EN DE
 210           LD   A,(HL)            ;COGE SIGUIENTE CARAC
TER
 220           SUB  "0"               ;LO CONVIERTE A NUMER
ICO
 230           INC  HL                ;INCREMENTA PUNTERO C
ADENA
 240           PUSH HL                ;LO GUARDA
 250           LD   L,A
 260           LD   H,0               ;PONE NUMERO EN HL
 270           ADD  HL,DE             ;VALOR BIN=VALOR BIN+
NUMERO
 280           POP  DE                ;PUNTERO CADENA EN DE
 290           DJNZ ICLP              ;REPITE, SI QUEDA CAD
ENA
 300           LD   (VARPTR),HL       ;VALOR BIN EN VARPTR
 310           LD   A,2
 320           LD   (VARTYP),A        ;VARTYP=ENTERO
 330           RET                    ;A BASIC
 340           END
```

CAPITULO 6

El Procesador de Video (VDP)

El procesador de pantalla de video (VDP) enlaza con la CPU a traves de un bus de datos bidireccional de 8 bits, 3 líneas de control y una interrupción. Pueden realizarse 4 operaciones: escritura de datos a la VRAM, lectura de datos desde la VRAM, escritura a los registros de control del VDP y lectura desde el registro de estado del VDP. Cada una de estas operaciones requiere una o más transferencias de datos mediante el enlace del bus de datos VDP/CPU, dependiendo dichas transferencias del estado de las 3 líneas de control. La CPU puede comunicarse con el VDP en modo asíncrono al finalizar cada ciclo de barrido de pantalla. El VDP tiene acceso a la VRAM, incluso durante un ciclo de barrido.

LAS LINEAS DE CONTROL

Las tres líneas de control (CSW, CSR y MODO) controlan la interpretación que hace el VDP de las transferencias de datos. CSV está activa (nivel bajo) cuando se realiza una transferencia de datos desde la CPU al VDP. CSR está activa (nivel bajo) cuando la CPU lee del VDP. CSW y CSR nunca deben de estar activas simultáneamente.

MODO determina la fuente o destino de la operación de lectura o escritura de datos, y en el sistema MSX suele estar conectada con la línea de dirección 0 de la CPU. Veáse la figura 6.1 para tener un resumen de las operaciones de enlace CPU/VDP. Estas operaciones se describen a continuación.

ESCRITURA DE LA CPU A UN REGISTRO DEL VDP

El VDP tiene 8 registros de control de escritura, y un registro de estado de lectura.

Cada uno de los 8 registros de escritura puede cargarse solamente mediante dos transferencias de datos de 8 bits, desde la CPU (el estado de las líneas de control necesarias para ésta y las demás operaciones pueden encontrarse en la tabla 6.1. El primer byte a enviar es el dato a transferir, y el segundo byte determina el registro de destino (con valor decimal comprendido entre 0 y 7, y con el bit más significativo a nivel 1, para distinguir la operación como escritura de registros, en oposición a una operación de modificación de direcciones de la VRAM).

Tabla 6.1 Transferencias de datos CPU/VDP

OPERACION	7	6	5	4	3	2	1	0	$\overline{CSW}$	$\overline{CSR}$	MODE
Escritura a registro del VDP											
Byte 1: dato a escribir	D7	D6	D5	D4	D3	D2	D1	D0	0	1	1
Byte 2: selección de registro	1	0	0	0	0	RS2	RS1	RS0	0	1	1
Escritura a VRAM											
Byte 1: dirección	A7	A6	A5	A4	A3	A2	A1	A0	0	1	1
Byte 2: dirección	0	1	A13	A12	A11	A10	A9	A8	0	1	1
Byte 3: dato a escribir	D7	D6	D5	D4	D3	D2	D1	D0	0	1	0
Lectura de registro del VDP											
Byte 1: dato a leer	D7	D6	D5	D4	D3	D2	D1	D0	1	0	1
Lectura de la VRAM											
Byte 1: dirección	A7	A6	A5	A4	A3	A2	A1	A0	0	1	1
Byte 2: dirección	0	0	A13	A12	A11	A10	A9	A8	0	1	1
Byte 3: dato a leer	D7	D6	D5	D4	D3	D2	D1	D0	1	0	0

Para volver a escribir datos en los registros internos, después de haber transferido un byte de datos, es necesario leer el registro de estado del VDP con el fin de reinicializar los dispositivos lógicos de enlace. Esta situación se produce frecuentemente en un medio dirigido por interrrupciones, como es el sistema MSX. En general, siempre que se dude del estado de los parámetros de lectura/escritura del VDP, debe utilizarse este procedimiento.

ESCRITURA DE LA CPU A LA VRAM

La transferencia de datos de la CPU a la VRAM utiliza un registro de direcciones de 14 bits que se incrementa automáticamente. Las transferencias de los dos primeros bytes de escritura a la VRAM, se emplean para inicializar este registro, y la escritura secuencial producida a continuación precisa solamente de la transferencia de un byte, puesto que este registro ya ha sido inicializado y se incrementa después de cada operación de escritura.

LECTURA DEL REGISTRO DE ESTADO DEL VDP POR LA CPU

La CPU puede leer el registro de estado del VDP utilizando solamente una transferencia de un byte. MODO está activo (nivel alto) para la transferencia, y CSR se emplea para señalizar que se precisa una operación de lectura.

LECTURA DE LA VRAM POR LA CPU

La CPU lee la VRAM en forma análoga a como la escribe, utilizando el registro de direcciones de incremento automático.

Como la CPU interactúa con la VRAM a través del VDP, es obvio que las transferencias de datos solamente pueden ocurrir cuando la VDP no está ocupada controlando la salida de video; por lo tanto, el tiempo necesario para que la CPU transfiera un byte de datos a la VRAM, varía entre 2 y 8 microsegundos, dependiendo de que la VDP esté ocupada con el refresco de memoria o con la pantalla. Los tiempos aproximados se encuentran en la tabla 6.2.

LOS REGISTROS DEL VDP

De los 8 registros que sólo admiten escritura, los numerados con 0 y 1 contienen banderas que controlan varias funciones y modos del VDP. Los registros del 2 al 6 contienen valores que especifican las direcciones de comienzo de los distintos subbloques de la VRAM, y el registro 7 se utiliza para definir el color de fondo en todos los modos y los colores de texto en el modo de 40 columnas. La descripción detallada de dichos registros se expone a continuación.

REGISTRO 0

Los 2 bits menos significativos de este registro son bits de control. Los bits restantes están reservados para una expansión futura, y deben tener nivel 0.

Bit 0: activación/desactivación de VDP externo (1=activación).

Tabla 6.2 Tiempos de acceso a la VRAM

CONDICION	MODO	RETARDO DEL VDP	TIEMPO DE ESPERA PARA UNA VENTANA DE ACCESO	TIEMPO TOTAL
Pantalla activa	Texto	2 us	0-1.1 us	2-3.1 us
Pantalla activa	Gráficos I y II	2 us	0-5.95 us	2-8 us
Pantalla activa	Multicolor	2 us	0-1.5 us	2-3.5 us
Registro 1 bit en blanco 0	Todos 2	2 us	0 us	2 us
4300 us después de la interrupción	Todos 2	2 us	0 us	2 us

Bit 1: tercer bit de modo, explicado a continuación.

REGISTRO 1 (8 BITS DE CONTROL DEL VDP)

Bit 7: selección de RAM 4/16K (siempre a nivel 1, en el sistema MSX).

Bit 6: activación/desactivación de color (1=activación de pantalla). La desactivación de color hace que la pantalla quede oscura.

Bit 5: activación de interrupciones (1=activación, 0=desactivación)

Bit 4: el primer bit de modo.

Bit 3: el segundo bit de modo, estando el tercero en el registro 0 Estos bits definen el modo de pantalla de acuerdo con la tabla:

M1	M2	M3	
0	0	0	Modo de Gráficos I
0	0	1	Modo de Gráficos II
0	1	0	Modo Multicolor
1	0	0	Modo de Texto

Bit 2: reservado para expanxión futura, debe ser 0.

Bit 1: selección de tamaño de sprite: 0 selecciona sprites de 8*8, 1 selecciona sprites de 16*16.

Bit 0: selecciona la opción de ampliación de sprites: 0 selecciona sprites normales, 1 selecciona sprites ampliados.

REGISTRO 2

Los 4 bits menos significativos del registro 2 forman los 4 bits superiores de la dirección de la tabla de nombres, que consta de 14 bits. Por lo tanto, la direccion de comienzo de la tabla de nombres es igual a (registro 2)*400H.

REGISTRO 3

El registro 3 define la dirección de comienzo en la VRAM de la tabla de color. El contenido de este registro constituye los 8 bits superiores de la dirección de 14 bits. Por lo tanto, la dirección puede calcularse haciendo (registro 3)*40H.

REGISTRO 4

Los tres bits menos significativos del registro 4 definen el comienzo del subbloque generador de patrones, texto o multicolor, formando los tres bits superiores de dicha dirección. Es decir, que la dirección del subbloque se obtiene mediante (registro 4)*800H.

REGISTRO 5

Los 7 bits inferiores de este registro conforman los 7 bits superiores de la tabla de atributos de sprite. Por tanto, la dirección es igual a (registro 5)*80H.

REGISTRO 6

El registro 6 define la dirección de comienzo en la VRAM de la tabla del generador de patrones de sprite. Esta dirección es (registro 6)*800H.

REGISTRO 7

Los 4 bits más significativos del registro 7 contienen el código del color 1 en el modo de texto, y los 4 bits inferiores, el código de color del color 0 en el modo de texto, y del de fondo, en los restantes modos.

REGISTRO DE ESTADO

El VDP tienen un único registro de estado de 8 bits, al que puede acceder la CPU. Este registro contiene la bandera de interrupción, de colisión de sprites, del quinto sprite y el número del quinto sprite (si existe). El formato del registro de estado se describe a continuación.

El registro de estado se puede leer en cualquier momento. Sin embargo, la lectura del registro de estado sin sincronía con la interrupción de barrido pondrá a 0 la bandera de cuadro, y puede hacer que se salten algunas interrupciones. Para evitar este problema, es aconsejable leer la copia de este registro que el sistema operativo guarda en la RAM del sistema (veáse el capítulo 5).

Bit 7: la bandera de interrupción. Esta bandera se pone a nivel 1 al terminar el barrido de la última línea de la pantalla (activa). Este nivel determina la activación de la patilla de interrupciones, si la bandera de activación de interrupciones, en el registro 0, está también a nivel 1. La bandera se pone a 0 mediante la lectura del registro de estado, por lo tanto, es preciso leer este registro al final de cada retorno de cuadro para reponer la bandera de interrupciones y reactivarla para el próximo cuadro.

Bit 5: la bandera de coincidencia de sprites. Esta bandera adquiere el nivel 1 siempre que dos sprites se encuentren en colisión (es decir, cuando tengan uno o más puntos superpuestos). Los sprites transparentes también son considerados, lo mismo que aquéllos que estén parcial o totalmente fuera de la pantalla. Sin embargo, aquellos sprites que se encuentren fuera del puntero que señala el fin de la tabla de atributos de sprite (DOH), no son considerados. La bandera se anula cuando se lee el registro de estado.

Bit 6: la bandera del quinto sprite. El bit 6 del registro de estado se pone a nivel 1 siempre que haya 5 ó más sprites en una línea horizontal. Se anula tras la lectura del registro de estado. Cuando la bandera adquiere el nivel 1, el número del quinto sprite perverso se sitúa en los 5 bits inferiores del registro de estado, permitiendo al usuario mover el sprite antes del próximo cuadro, para asegurarse de que todos los sprites se muestran convenientemente.

MODOS DE PANTALLA

El VDP genera una imagen que puede ser considerada como un cierto número de planos superpuestos. El orden de prioridad de dicho planos va desde el frente hacia atrás, por lo que si dos objetos coinciden en la misma posición, pero en planos diferentes, el objeto en el plano de prioridad más alta ocultará los objetos en cualquier plano de inferior prioridad. Por tanto, un sprite en el plano 0 aparecerá delante de los del resto de la pantalla. De delante a atrás, los planos son: los 32 planos de sprite, seguidos por el plano patrón, seguido por el plano de fondo, seguido por los planos de cualquier VDP externo que se conecte. Sin embargo, se desconoce la intención de las compañías de MSX de introducir aparatos con más de un VDP.

El plano de fondo es un único color, y se utiliza para mostrar las áreas de borde, y es el color por omisión del plano patrón. Cuando se cambia a transparente, automáticamente se convierte en negro, salvo que esté funcionando un VDP externo.

Los 32 planos de sprite se encuentran sobre el plano patrón. El sprite 1 presenta la prioridad inferior, y el sprite 0 la superior. Los sprites están inactivos en el modo de texto, en los restantes modos permanecen transparentes por omisión. Sus posiciones pueden definirse coordenada a coordenada, permitiendo su suave movimiento. La forma de almacenamiento y la apariencia del plano patrón, difiere en los distintos modos de pantalla, y será desarrollado separadamente.

Se dispone de 16 colores en todos los modos, con los códigos:

CODIGO	COLOR
0	Transparente
1	Negro
2	Verde
3	Verde claro
4	Azul oscuro
5	Azul claro
6	Rojo oscuro
7	Cyan
8	Rojo
9	Rojo claro
10	Amarillo oscuro
11	Amarillo claro
12	Verde oscuro
13	Magenta
14	Gris
15	Blanco

MODO GRAFICO I

En el modo gráfico I, el subbloque de RAM que conforma la tabla de nombres, está compuesta de 768 bytes, dispuestos en 24 filas de 32 caracteres. Por lo tanto, los primeros 32 valores en el mapa de la tabla, forman la primera línea de pantalla, los siguientes 32 la

segunda línea, y así, sucesivamente. Cada elemento de la tabla hace referencia a una de las 256 definiciones de patrón almacenadas en la tabla de patrones generadores. Cada uno de estos 256 elementos, consiste en 8 bytes, obteniéndose 8*8 bits, por lo que la tabla necesita 2048 bytes de VRAM en total. En cada elemento de 8 bytes de la tabla de patrones generadores, un bit a nivel 1 se muestra como el color 1, y un bit a nivel 0, se muestra como el color 0. Los colores de cada patrón se toman de la tabla de color, que en el modo gráfico I tiene 32 bytes de longitud. Cada elemento define un color de primer plano (color 1) y un color de fondo (color 0) de 8 patrones en la tabla de patrones generadores. El primer elemento en la tabla de color define los colores de los patrones 0 a 7 de la tabla de patrones generadores, el segundo elemento los colores de los patrones 8 a 15, etc. Los elementos de la tabla de color se consideran como dos nibbles, los 4 bits más significativos definen el color de primer plano, y el nibble menos significativo el color de fondo. De lo anterior puede deducirse que la implementación completa de este modo requiere 2848 bytes de VRAM. Sin embargo, si no se precisan las 256 definiciones de patrón totales, es posible superponer las tablas.

MODO GRAFICO II

En el modo gráfico II la tabla de nombres está dispuesta de la misma manera que en el modo gráfico I, excepto que cada elemento en la tabla de nombres puede tener su propia y única definición en la tabla de patrones generadores. Esto se consigue dividiendo dicha tabla en tres bloques de 256 definiciones; cada elemento de los primeros 256 bytes de la tabla de nombres (es decir, el tercio superior de la pantalla) hacen referencia a las definiciones 0 a 255 de la tabla de patrones generadores y los valores del segundo y tercer bloque de 256 bytes de la tabla de nombres se refieren a las definiciones 256 a 511 y 512 a 767, respectivamente.

También es posible definir los colores de primer plano y fondo de cada línea horizontal de cada definición de patrón. La tabla de color consiste en 768 elementos de 8 bytes. Cada uno de estos bytes se corresponde con un byte de la tabla de patrones generadores. Los cuatro bits más significativos de cada byte de la tabla de color, definen el color de primer plano del byte correspondiente de la tabla de patrones generadores, y los cuatro bits menos significativos, el color de fondo.

Como puede apreciarse, mediante un manejo cuidadoso de los valores de la tabla de nombres en este modo gráfico, dicha tabla puede considerarse reflejada bit a bit en la memoria (bit-mapped). Si copiamos idénticos valores en los tres bloques de las tablas de patrones generadores y color, el modo gráfico II se puede utilizar en forma similar al modo gráfico I, pero con mejor definición del color.

MODO MULTICOLOR

El modo multicolor permite mostrar una pantalla de 64*48 puntos, sin limitación de color. Cada uno de estos puntos "gruesos" está formado

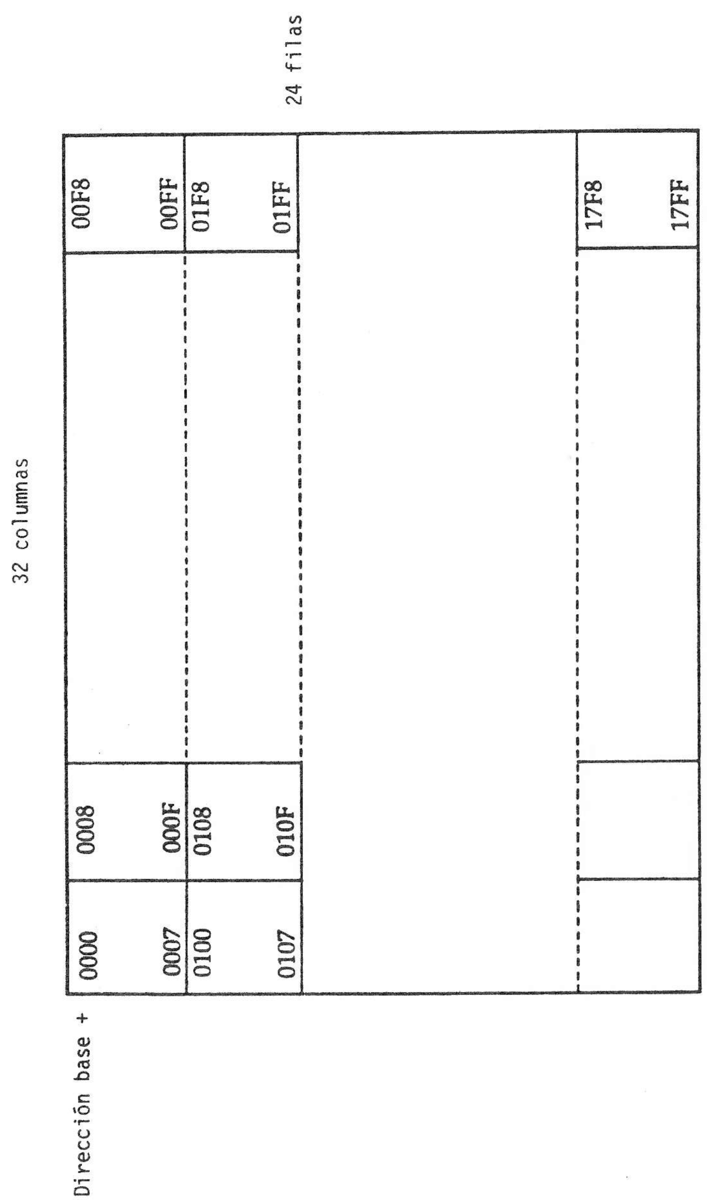


Figura 6.1 Mapa de memoria de la pantalla de gráficos II

por un bloque de 4*4 puntos. El color de cada cuadrado puede ser cualquiera de los quince disponibles en el VDP.

La tabla de nombres del modo multicolor es similar, en su configuración, a la de los dos modos gráficos, aunque el nombre no apunte a ningún elemento de la tabla de color. El color se obtiene de los elementos de la tabla de patrones generadores.

Cada valor en la tabla de nombres apunta a un bloque de 8 bytes de la tabla de patrones generadores. Solamente dos bytes de este bloque se utilizan para definir la imagen de pantalla. Estos dos bytes, descompuestos en cuatro nibbles, definen un patrón multicolor de 8*8 puntos. El nibble de orden superior del primer byte define el color del cuadrado de 4*4 puntos en la esquina superior izquierda del patrón multicolor, el nibble inferior define el color de la esquina superior derecha, y el segundo byte realiza la misma función para el color de los cuadrados inferior izquierdo y derecho, respectivamente. Los elementos de las filas 0, 4, 8, 12, 16 y 20 de la tabla de nombres, utilizan los primeros dos bytes del bloque de 8 bytes de la tabla de patrones generadores, los elementos de las filas 1, 5, 9, 13, 17 y 21 emplean los segundos dos bytes, y así, sucesivamente.

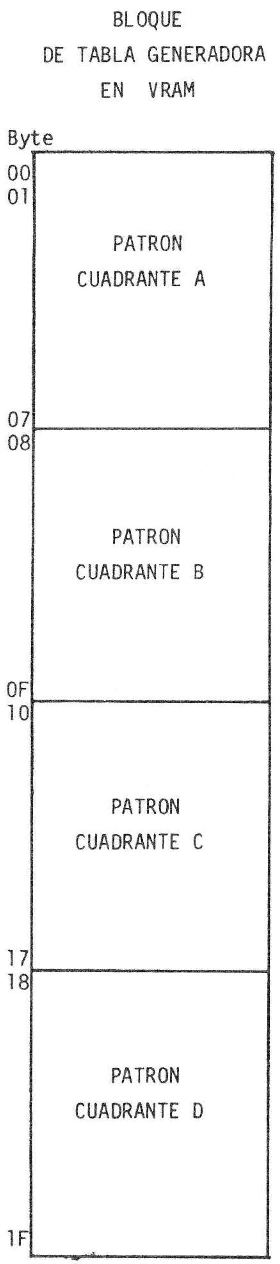
MODO DE TEXTO

En el modo de texto, la pantalla puede considerarse como una retícula de 40 caracteres de ancho por 24 líneas de altura. Cada elemento de esta retícula tiene 6 puntos de anchura por 8 puntos de altura. Las tablas empleadas para generar el plano patrón son: la tabla de nombres y la tabla de patrones generadores, pudiendo existir hasta 256 patrones diferentes definidos simultáneamente. La tabla de nombres ubica las definiciones en cada una de las 960 celdas de patrón del plano patrón. Los sprites no pueden emplearse en este modo. La tabla de patrones generadores es idéntica a la del modo gráfico I, excepto que, como cada celda de patrón tiene solamente 6 puntos de anchura, los dos bits menos significativos de cada byte de la tabla de patrones generadores se ignoran. Los colores de primer plano y fondo se fijan mediante el valor del registro 7 del VDP. El nibble superior define el color de primer plano, y el inferior, el color de fondo y de borde.

SPRITES (FIGURAS MOVILES)

La pantalla de video puede contener hasta 32 sprites en los planos de máxima prioridad. Sus posiciones se definen desde la esquina superior izquierda del patrón de sprite, como esta posición puede desplazarse punto a punto, es muy fácil mover suave y rápidamente las figuras móviles. Cada uno de los planos de sprite es totalmente transparente, con la excepción del sprite mismo. Los sprites no están activados en el modo de texto de 40 columnas.

Las tablas de la VRAM necesarias para la utilización de los sprites son: la tabla de atributos de sprite y la tabla de patrones generadores de sprite. Estas tablas son parecidas a sus equivalentes en el plano



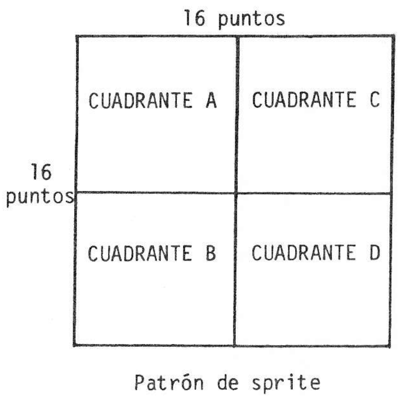


Figura 6.2 Mapa de sprites de 16x16

patrón, puesto que la tabla de atributos de sprite define la posición del sprite, y la tabla de patrones de sprite contiene una definición de su aspecto.

Cada uno de los 32 elementos de la tabla de atributos de sprite contiene 4 bytes, que contienen: el múmero de puntos desde su esquina superior izquierda hasta el borde superior de la pantalla (su posición vertical), seguido por su posición horizontal (es decir, el número de puntos desde el borde lateral izquierdo de la pantalla), seguido por su nombre, esto es, su definición en la tabla de patrones generadores de sprite, y, finalmente, su "rótulo", del cual los cuatro bits menos significativos definen su color, y el bit más significativo es el bit previo de reloj. Este último hace que el sprite aparezca 32 puntos a la izquierda de su posición, como se define por el byte 2 del elemento de la tabla de atributos de sprite.

Cuando un sprite se sitúa en las coordenadas (0,0), toca contra el borde superior de la pantalla. En muchas aplicaciones, se hace preciso que el sprite aparezca a partir de los bordes de la pantalla. Para realizar este efecto, existe un método, diferente para cada eje, que se explica a continuación:

En primer lugar, la posición vertical del sprite se considera con signo en los valores comprendidos entre -31 y 0 (en complemento de 2), lo que permite al sprite aparecer a partir del borde superior. Como puede suponerse, debido a que la resolución horizontal de la pantalla es de 256 puntos, este método no pude utilizarse para hacer aparecer los sprites desde el borde izquierdo; para ello se dispone del bit previo de reloj en el byte 4 de cada elemento de la tabla de atributos de sprite.

La tabla de patrones de sprite dispone de un máximo de 256 elementos de 8 bytes. El tercer byte de cada elemento de la tabla de atributos de sprite define qué bloque de la tabla de patrones se va a utilizar para generar el sprite correspondiente a dicho plano. Cuando se utilizan sprites de 16*16 puntos, este valor se divide por cuatro, apuntando a un bloque de 32 bytes de la tabla de patrones generadores de sprite.

Por último, hay que comentar que sólamente se pueden mostrar un maximo de cuatro sprites en cada línea horizontal. Si esta regla no se respeta, únicamente se verán los cuatro sprites de mayor prioridad; asímismo, la bandera de quinto sprite se pondrá a nivel 1 en el registro de estado, y el múmero del quinto sprite se cargará en los cinco bits inferiores.

EL PROCESADOR DE VIDEO DENTRO DEL SISTEMA MSX

En el sistema MSX, puede accederse al VDP de dos formas diferentes. Por una parte, la ROM del sistema nos permite disponer de las rutinas necesarias para realizar la mayoría de las funciones básicas. En algunos casos, sin embargo, estas rutinas pueden ser inadecuadas en su función o en su velocidad de ejecución, siendo posible acceder al VDP directamente, puesto que las posicioens 6 y 7 de la ROM del sistema contienen direcciones de lectura y escritura del VDP.

La existencia de dos direcciones deriva de la línea de control MODO del VDP.

Con referencia a la tabla 6.1 podemos ver que para escribir a un registro del VDP, la línea MODO debe estar activa (nivel alto), mientras que para leer o escribir a la VRAM, la línea MODO debe estar a nivel bajo. Como la línea MODO del VDP está normalmente enlazada con una de las líneas de direcciones de la CPU, podríamos esperar que la dirección dada por las posiciones 6 y 7 fuera diferente solamente por la presencia o ausencia de un único bit. Si examinamos la ROM, encontramos que la posición 6 contiene el valor 98H, y la posición 7 el 99H, así, podemos deducir que la línea MODO está conectada con la línea de direccion cero. Las otras dos líneas de control del VDP, CSR y CSW, están conectadas mediante puertas lógicas a las líneas de control del Z-80 IORQ, RD y WR, de forma tal que su control se realice automáticamente. La siguiente rutina escribirá a la posición de la VRAM, especificada con una dirección de 14 bits contenida en HL, el dato del acumulador, ilustrando lo anteriormente expuesto.

```
  10 WRTVRM:   PUSH AF            ;GUARDA DATOS
  20           LD   A,(6)         ;COGE DIRECCION VDP M
ODE ACTIVO
  30           LD   C,A           ;EN C
  40           INC  C             ;MODE ACTIVO
  50           DI                 ;DESCONECTA INTERRUPC
ION
  60           OUT  (C),L         ;SALIDA BYTE INFERIOR
 DIRECCION
  70           SET  6,H           ;BIT 6 BYTE SUPERIOR
DIRECCION=1
  80           OUT  (C),H         ;LO ENVIA
  90           DEC  C             ;MODE INACTIVO
 100           POP  AF            ;RESTAURA DATOS
 110           OUT  (C),A         ;LO ENVIA
 120           EI                 ;ACTIVA INTERRUPCION
 130           RET                ;RETORNA
```

El sistema opeativo dispone de rutinas rápidas y óptimas para la mayoría de las funciones que pueden precisarse, y que se relacionan a continuacion.

DIRECCION	FUNCION
0041H	Desactiva la pantalla. Esta rutina no requiere parámetros, pero modifica los pares de registros AF y BC.
0044H	Activa la pantalla, no requiere parámetros y modifica los pares de registros AF y BC.
0047H	Escribe el dato contenido en B, al registro especificado por C, modificando los pares de registros AF y BC.
004AH	Lee un byte de la VRAM en la dirección especificada por el par HL, almacenando el valor leido en el acumulador y modificando solamente el par AF.

004DH	Escribe el dato del acumulador en la dirección de la VRAM especificada por el contenido del par HL, modificando solamente el par AF.
0050H	Prepara el VDP para una lectura de la VRAM, acepta la dirección inicial de lectura en HL, y modifica el par AF.
0053H	Esta rutina es análoga a la anterior, pero preparando al VDP para una operación de escritura.
0056H	Esta rutina rellena el número de bytes de la VRAM, especificado por el par BC, empezando por la dirección contenida en el par HL con el dato pasado en el acumulador. Modifica los pares BC y AF.
0059H	Mueve un bloque de memoria desde la VRAM a la RAM, su funcionamiento es análogo a la rutina siguiente.
005CH	Mueve un bloque de memoria desde la memoria de la CPU a la VRAM. Acepta la dirección de origen en HL, la de destino en DE, y la longitud del bloque a mover en BC, modificando todos los registros.
005FH	Dispone el VDP en el modo de pantalla definido por la posición CAFH (0 texto 40, 1 gráficos I, 2 gráficos II, 3 multicolor). Esta rutina modifica todos los registros.
0062H	Cambia el color de la pantalla, tomando los parámetros de F3E9H (color de primer plano), F3EAH (color de fondo) y F3EBH (color de borde). Esta rutina modifica todos los registros.
0069H	Inicializa todos los sprites. Los patrones de sprite se ponen en blanco, los nombres de sprite se fijan con los números de plano de sprite, los colores adquieren el color de primer plano, y la posición vertical se iguala a 209. Esta rutina requiere que el modo de pantalla del VDP se sitúe en FCAFH y modifica todos los registros.
006CH	Inicializa el VDP en el modo de texto de 40 columnas. Esta rutina utiliza los valores del área de RAM del sistema desde F3B3H en adelante, y modifica todos los registros.
006FH	Inicializa el VDP en el modo de gráficos I, siendo semejante junto con las dos rutinas siguientes, a la rutina anterior.
0072H	Inicializa el VDP en el modo gráfico II.
0075H	Inicializa el VDP en el modo multicolor.

En circunstancias normales, el acceso directo al VDP por el usuario, es raramente necesario.

PROGRAMACION DEL VDP: CONSEJOS Y SUGERENCIAS

En los modos de gráficos I y II, que son los de mayor interés para el programador, en general, la apariencia del plano patrón está por completo a disposición del programador, puesto que todas las definiciones de caracteres se toman de la RAM. Es aconsejable, sin embargo, seguir unas directrices en la producción de las definiciones de caracteres particulares. Por ejemplo, en los aparatos MSX está claro que conviene situar en VRAM por lo menos las letras mayúsculas y los números de forma que sea posible escribir el valor ASCII de cualquier número o letra en la tabla de nombres para visualizarlos. La definición de "A" (valor ASCII 65) se encuentra en la VRAM en una dirección que puede calcularse haciendo: (dirección de base de la tabla de patrones generadores)+(65*8)

La forma más sencilla para conseguirlo es tomar las definiciones básicas de su posición en ROM, y después redefinir aquéllas que uno vaya a necesitar. Esto se hace por el sistema operativo, en el modo de gráficos I, al ejecutar una instrucción SCREEN 1; sin embargo, cuando esta instrucción se emplea para pasar al modo de gráficos II, el sistema operativo dispone el VDP con las 768 definiciones disponibles en blanco. Se hace necesario, por tanto, localizar las definiciones de caracteres standard en la ROM del sistema. Esta posición difiere de una máquina a otra, por lo que utilizaremos el siguiente programa para localizar la letra "A" en la ROM del MSX.

```
10 FOR N%=&H0 TO &H7FFF
20 A%=PEEK(N%):READ B%
30 IF A%<>B% THEN RESTORE:Q%=0 ELSE Q%=Q%+1
40 IF Q%=8 THEN PRINT HEX$(N%-8):END
50 NEXT N%
60 END
70 DATA 32,80,136,136,248,136,136,0
```

Los siguientes programas, en código máquina y BASIC, ilustran sobre muchos de los puntos a considerar cuando se emplea el VDP. También nos permiten disponer de un potente programa para la definición de caracteres y sprites, a utilizar especialmente en el modo de gráficos II (pero puede también utilizarse en los otros modos).

```
 10 WRTVDP:   EQU  #0047              ;TABLA DE LLAMADAS A BIOS
 20 RDVRM:    EQU  #004A
 30 WRTVRM:   EQU  #004D
 40 FILVRM:   EQU  #0056
 50 LDIRVM:   EQU  #005C
 60 LDIRMV:   EQU  #0059
 70 CHGET:    EQU  #009F
 80 GTSTCK:   EQU  #00D5
 90 GTTRIG:   EQU  #00D8
100 RDVDP:    EQU  #013E
110 SNSMAT:   EQU  #0141              ;FIN TABLA BIOS
120 ;DEFINICION DE COLOR/CARACTER MODO GRAFICO II
130 NAMTAB:   EQU  #1800
```



```
140 COLTAB:   EQU  #2000              ;DIRECCIONES VRAM PARA GM2
150 PATTAB:   EQU  #0000
160 SATTAB:   EQU  #1B00
170 SPTTAB:   EQU  #3800              ;FIN DIRECCIONES VRAM
180 RNAMTB:   EQU  #0006
190 RCOLTB:   EQU  #00FF              ;VALOR REGISTROS VDP MODO GRAFICO II
200 RPTTAB:   EQU  #0003
210 RSATAB:   EQU  #0036
220 RSPTAB:   EQU  #0007              ;FIN VALORES
230 CHARDT:   EQU  #1CBF              ;POSICION DE CARACTERES EN ROM, PUEDE DIF
ERIR
240           ORG  #9000
250 REGDAT:   DEFB 2,#00E0,RNAMTB,RCOLTB,RPTTAB,RSATAB,RSPTAB,6
260 START:    LD   E,8
270           LD   IX,REGDAT          ;PUNTERO DE DATOS EN IX
280           LD   D,0                ;CARGA DATOS
290 SVDPLP:   LD   B,(IX+0)           ;DE REGISTROS EN B
300           LD   C,D                ;NO. DE REGISTRO EN C
310           CALL WRTVDP             ;EN EL VDP
320           INC  IX                 ;INCREMENTA PUNTERO DATOS
330           INC  D                  ;INCREMENTA NO. REGISTRO
335           DEC  E
340           JR   NZ,SVDPLP          ;8 REGISTROS
350           LD   HL,CHARDT          ;CONSIGUE DIRECCION CARACTERES
360           LD   DE,PATTAB+#1100    ;DE SU DIRECCION EN VRAM
370           LD   BC,60*8            ;60 CARACTERES
380           CALL LDIRVM
390           LD   BC,#0800
400           LD   HL,COLTAB+#1000
410           LD   A,#00F1            ;PREPARA COLORES
420           CALL FILVRM
430           LD   BC,#0800
440           LD   HL,COLTAB
450           LD   A,#001F
460           CALL FILVRM             ;PREPARA COLORES
470           LD   BC,#0800
480           LD   HL,COLTAB+#0800
490           LD   A,#0081
500           CALL FILVRM             ;MAS COLORES
510           LD   BC,768
520           LD   HL,NAMTAB
530           LD   A,32
540           CALL FILVRM             ;BORRA PANTALLA
550           LD   HL,P2DAT
560           LD   DE,PATTAB+#0800
570           LD   BC,24
580           CALL LDIRVM             ;DEFINE CARACTERES 0..2 EN BLOQUE 2
590           LD   HL,P2DAT+16
600           LD   DE,SPTTAB
610           LD   BC,8
620           CALL LDIRVM             ;DEFINE SPRITE 0
630           CALL FUNCPT             ;IMPRIME COLUMNAS FUNCION
640           CALL FTTXRM             ;TRANSFIERE DATOS VRAM A RAM
650 MLP:      CALL ARRYPT             ;IMPRIME MATRIZ DEFINICION
660           CALL SPON               ;IMPRIME CURSOR
670           CALL JOY                ;LEE MANDO
680           CALL FUNC               ;IMPRIME MENSAJES FUNCION
690           HALT                    ;ESPERA FIN CUADRO
700           CALL DOFUNC             ;HACE FUNCIONES
710 NDFNC:    HALT                    ;ESPERA FIN CUADRO
720           HALT
730           CALL CHART              ;TRANSFIERE DEFINICION
740           CALL CSPT               ;DE MATRIZ A RAM
750           CALL COLPT              ;E IMPRIME MENSAJE ESTADO
760           CALL BANKPT
770           CALL LPTXRM             ;TRANSFIERE DATOS VRAM A RAM
780           HALT                    ;ESPERA FIN CUADRO
790           JR   MLP                ;VUELVE Y REPITE
800           RET
```

```
810 SPY:       DEFB 0                ;Y CURSOR
820 SPX:       DEFB 0                ;X CURSOR
830 ARRYPT:    LD   C,8              ;RUTINA IMPRIME MATRIZ DEFINICION
840            LD   HL,NAMTAB+259    ;DIRECCION TABLA NOMBRES + DESPLAZAMIENTO
850            LD   IX,HARRY1        ;DIRECCION CONJUNTO EN IX
860 APTOLP:    LD   B,8              ;BUCLE INTERIOR DE 8
870 APTILP:    LD   A,(IX+0)         ;CONSIGUE DATOS CONJUNTO
880            PUSH BC               ;GUARDA CONTADORES
890            CALL WRTVRM           ;ESCRIBE DATOS CONJUNTO A PANTALLA
900            POP  BC               ;REPONE CONTADORES
910            INC  HL               ;INCREMENTA PUNTERO PANTALLA
920            INC  IX               ;INCREMENTA PUNTERO CONJUNTO
930            DJNZ APTILP           ;FIN BUCLE INTERIOR
940            LD   DE,24            ;LONGITUD LINEA 8
950            ADD  HL,DE            ;SE SUMA A PUNTERO PANTALLA
960            DEC  C                ;DECREMENTA CUENTA BUCLE EXTERIOR
970            JR   NZ,APTOLP        ;8 VECES
980            RET
990 FUNCPT:    LD   HL,NAMTAB+258    ;DIRECCION DE COLUMNA IZQUIERDA CONJUNTO
1000           CALL ROWOF8           ;IMPRIME COLUMNA
1010           LD   HL,NAMTAB+267    ;DERECHA DEL CONJUNTO
1020           CALL ROWOF8           ;IMPRIME COLUMNA
1030           RET
1040 ROWOF8:   LD   B,8              ;CUENTA 8
1050 RO8LP:    PUSH BC               ;GUARDA CONTADOR
1060           PUSH HL               ;GUARDA DIRECCION
1070           LD   A,2              ;CARACTER2
1080           CALL WRTVRM           ;A DIRECCION PANTALLA
1090           POP  HL               ;REPONE DIRECCION PANTALLA
1100           LD   DE,32            ;SUMA LONGITUD LINEA
1110           ADD  HL,DE            ;A DIRECCION PANTALLA
1120           POP  BC               ;REPONE CONTADOR
1130           DJNZ RO8LP            ;8 VECES
1140           RET
1150 SPON:     LD   HL,SATTAB        ;RUTINA PARA SITUAR CURSOR
1160           LD   A,(SPY)          ;TOMA COORDENADA Y CURSOR
1170           SLA  A
1180           SLA  A                ; MULTIPLICA POR 8
1190           SLA  A                ; PORQUE LA POSICION DE SPRITE=CHARPOS*8
1200           ADD  A,63             ;SUPERIOR IZQUIERDA MATRIZ,DESPLAZAMIENTO
1210           CALL WRTVRM           ;ESCRIBE A TABLA ATRIBUTOS SPRITE
1220           INC  HL               ;INCREMENTA PUNTERO VRAM
1230           LD   A,(SPX)          ;TOMA SPRITE X
1240           SLA  A
1250           SLA  A
1260           SLA  A                ;*8
1270           ADD  A,16             ;SUMA X DESPLAZAMIENTO
1280           CALL WRTVRM           ;A TABLA ATRIBUTOS SPRITE
1290           INC  HL               ;APUNTA NOMBRE SPRITE
1300           LD   A,0              ;NOMBRE ES 0
1310           CALL WRTVRM           ;ESCRIBE ATRIBUTOS SPRITE
1320           INC  HL               ;INCREMENTA PUNTERO
1330           LD   A,15             ;SPRITE BLANCO
1340           CALL WRTVRM           ;A ATRIBUTOS
1350           RET
1360 P2DAT:    DEFB 255,129,129,129,129,129,129,255;DEFINICIONES DE
1370           DEFB 255,255,255,255,255,255,255,255;CARACTERES
1380           DEFB 255,195,165,153,153,165,195,255;0-2 Y SPRITE 0
1390 HARRY1:   DEFB 0,0,0,0,0,0,0,0
1400           DEFB 0,0,0,0,0,0,0,0
1410           DEFB 0,0,0,0,0,0,0,0
1420           DEFB 0,0,0,0,0,0,0,0
1430           DEFB 0,0,0,0,0,0,0,0
1440           DEFB 0,0,0,0,0,0,0,0
1450           DEFB 0,0,0,0,0,0,0,0
1460           DEFB 0,0,0,0,0,0,0,0  ;CONJUNTO DE CARACTER
1470 HARRY2:   DEFB 0,0,0,0,0,0,0,0
1480           DEFB 0,0,0,0,0,0,0,0
1490           DEFB 0,0,0,0,0,0,0,0
1500           DEFB 0,0,0,0,0,0,0,0
1510           DEFB 0,0,0,0,0,0,0,0
1520           DEFB 0,0,0,0,0,0,0,0
```



```
1530           DEFB 0,0,0,0,0,0,0,0
1540           DEFB 0,0,0,0,0,0,0,0  ;CONJUNTO TEMPORAL PARA TRANSFORMACIONES
1550 JOY:      LD   A,0              ;PARA LEER JOY(0) TECLAS CURSOR
1560           CALL GTSTCK           ;CAMBIESE PARA LEER MANDOS
1570           LD   HL,(SPY)         ;X,Y CURSOR EN HL
1580           CP   1                ;MANDO ADELANTE?
1590           CALL Z,DEY            ;SI,DECREMENTA Y
1600           CP   3                ;MANDO DERECHA?
1610           CALL Z,INX            ;SI,INCREMENTA X
1620           CP   5                ;MANDO ABAJO?
1630           CALL Z,INY            ;SI,INCREMENTA Y
1640           CP   7                ;MANDO IZQUIEDA?
1650           CALL Z,DEX            ;SI,DECREMENTA X
1660           LD   A,L              ;CONSIGUE Y
1670           CP   8
1680           JR   NZ,JOY1
1690           LD   L,0              ;SI ES PRECISO ROTALO
1700 JOY1:     CP   255
1710           JR   NZ,JOY2
1720           LD   L,7
1730 JOY2:     LD   A,H              ;MAS ROTACION
1740           CP   10
1750           JR   NZ,JOY3
1760           LD   H,0              ;SI ES PRECISO ROTA X
1770 JOY3:     CP   255
1780           JR   NZ,JOY4
1790           LD   H,9
1800 JOY4:     LD   (SPY),HL         ;ALMACENA X,Y CURSOR
1810           RET                   ;RETORNO
1820 INY:      INC  L
1830           RET
1840 DEY:      DEC  L
1850           RET
1860 INX:      INC  H
1870           RET
1880 DEX:      DEC  H
1890           RET
1900 EXIT:     LD   A,0              ;SALIDA RUTINA
1910           LD   (BASFNC),A       ;SALIDA ES FUNCION 0
1920           POP  BC               ;TOMA DIRECCION RETORNO
1930           RET                   ;RETORNO A BASIC
1940 GETMGS:   LD   A,(MENU)         ;MENU 0/1
1950           LD   IY,MGSTAB        ;IY APUNTA A MENSAJES
1960           SLA  A                ;MENU ES 0/2
1970           LD   C,A
1980           LD   B,0
1990           ADD  IY,BC            ;IY=IY O IY=IY+2
2000           LD   A,(IY+0)         ;DIRECCION MENSAJES ORDEN BAJO
2010           LD   C,A              ;EN C
2020           LD   A,(IY+1)         ;ORDEN ALTO
2030           LD   B,A              ;EN B
2040           PUSH BC               ;TOMA MENSAJE TABLA DIRECCIONES
2050           POP  IX               ;EN IX
2060           RET
2070 DFUNC:    DEFB 0
2080 MGSTAB:   DEFW FNMSGS,FNMSG1    ;DIRECCIONES TABLAS DE MENSAJE
2090 MENU:     DEFB 0                ;NO. MENU
2100 FUNC:     CALL SPFNT            ;BORRA VIEJO MENSAJE
2110           LD   A,(SPX)          ;COGE SPRITE X
2120           CP   0                ;COLUMNA DE FUNCION DERECHA?
2130           JR   Z,FUNC1          ;SI,IMPRIME MENSAJE FUNCION
2140           CP   9                ;COLUMNA IZQUIERDA?
2150           RET  NZ               ;SI NO, RETORNA
2160           LD   A,8
2170 FUNC1:    LD   B,A
2180           LD   A,(SPY)
2190           ADD  A,B
2200           LD   B,A
2210           LD   (DFUNC),A        ;CALCULA Y GUARDA NO. FUNCION
2220           PUSH BC               ;GUARDA BC
2230           CALL GETMGS           ;COGE DIRECCION MENSAJE
2240           POP  BC               ;REPONE BC
```

```
2250            CALL MSGFND               ;ENCUENTRA MENSAJE EN TABLA
2260 FNPNT:     LD   HL,NAMTAB+515
2270 FNPNT1:    LD   A,(IX+0)
2280            CP   "$"
2290            RET  Z
2300            CALL WRTVRM
2310            INC  HL
2320            INC  IX
2330            JR   FNPNT1               ;IMPRIME MENSAJE
2340 SPPNT:     PUSH AF                   ;RUTINA IMPRESION ESPACIOS
2350            LD   IX,NOFUNC
2360            CALL FNPNT
2370            POP  AF
2380            RET
2390 MSGFND:    DEC  B                    ;ENCUENTRA MENSAJE B EN CADENA
2400            RET  M
2410 MSGFN1:    LD   A,(IX+0)             ;DIRECCIONADA POR IX
2420            INC  IX
2430            CP   "$"
2440            JR   NZ,MSGFN1
2450            JR   MSGFND
2460 NOFUNC:    DEFM "                   $";CADENA SIN FUNCION
2470 FNMSGS:    DEFM "SALIDA$CARGA$GUARDA$BORRA$";CADENA MENSAJE MENU 1
2480            DEFM "CARACTER$SPRITE$COLOR$"
2490            DEFM "SCROLL IZQ$SCROLL DER$"
2500            DEFM "SCROLL ARR$SCROLL ABA$"
2510            DEFM "COPIA$INVIERTE$SIMET VER$"
2520            DEFM "SIMET HOR$MENU 2$"
2530 FNMSG1:    DEFM "POSICION$JUEGO ALTERNO$AMPLIA SPRITE$";CADENA MENU 2
2540            DEFM "REDUCE SPRITE$SPRITE 16*16$"
2550            DEFM "SPRITE 8*8$NO IMPLEMENTADA$"
2560            DEFM "NO IMPLEMENTADA$NO IMPLEMENTADA$"
2570            DEFM "NO IMPLEMENTADA$NO IMPLEMENTADA$"
2580            DEFM "NO IMPLEMENTADA$NO IMPLEMENTADA$"
2590            DEFM "NO IMPLEMENTADA$NO IMPLEMENTADA$"
2600            DEFM "MENU 1$"
2610 EDIT:      LD   IX,HARRY1            ;IX PUNTERO DE CONJUNTO
2620            LD   A,(SPX)
2630            DEC  A
2640            LD   C,A
2650            LD   B,0
2660            ADD  IX,BC                ;SUMA DESPLAZAMIENTO X
2670            LD   A,(SPY)              ;COGE DESPLAZAMIENTO Y
2680            SLA  A
2690            SLA  A
2700            SLA  A                    ;8 VECES
2710            LD   C,A
2720            ADD  IX,BC                ;SUMA DESPLAZAMIENTO Y
2730            LD   A,(IX+0)
2740            INC  A
2750            AND  1
2760            LD   (IX+0),A             ;ALTERNA POSICION CONJUNTO
2770            RET
2780 DOFUNC:    LD   A,0
2790            CALL GTTRIG               ;ESTA PULSADA LA BARRA DE ESPACIO?
2800            CP   255
2810            RET  NZ                   ;SI NO, RETORNA
2820            LD   A,(SPX)
2830            CP   0
2840            JR   Z,DOFN1
2850            CP   9
2860            JR   NZ,EDIT              ;SI NO ESTA EN COLUMNA DE FUNCION
2870 DOFN1:     LD   IX,DFUNC
2880            LD   A,(MENU)
2890            SLA  A
2900            SLA  A
2910            SLA  A
2920            SLA  A
2930            ADD  A,(IX+0)
2940            SLA  A
2950            LD   C,A
2960            LD   B,0
```



```
2970            LD   IX,FNTABL
2980            ADD  IX,BC
2990            LD   A,(IX+0)             ;COGE DIRECCION DE
3000            LD   L,A                  ;TABLA DE SALTO
3010            LD   A,(IX+1)             ;Y LA PONE
3020            LD   H,A                  ;EN HL
3030            JP   (HL)                 ;Y LA EJECUTA
3040 FNTABL:    DEFW EXIT,LOAD,SAVE,CLR,CHAR,SPRIT;TABLA DE SALTO DE FUNCION
3050            DEFW COLR,LSCRL,RSCRL,USCRL
3060            DEFW DSCRL,COPY,INVT,FVER,FHOR,NXTMNU
3070            DEFW POSIT,BANKSW,MSP
3080            DEFW DMSP,SP16,SP8,NIMP,NIMP,NIMP,NIMP
3090            DEFW NIMP,NIMP,NIMP,NIMP,NIMP,NXTMNU
3100 NXTMNU:    LD   A,(MENU)             ;ALTERNA MENU
3110            INC  A
3120            AND  1
3130            LD   (MENU),A
3140 NIMP:      RET                       ;RUTINA FANTASMA CUANDO NO HAY FUNCIONES
3150 SPRIT:     LD   HL,SPTTAB            ;RUTINA SELECCION SPRITE
3160            LD   (CHSP),HL            ;A DEFINIR
3170            JR   CHAR1                ;FIJANDO DIRECCION SPRITE
3180 CHARF:     DEFB 0                    ;Y HACIENDO SELECCION CARACTER
3190 CHAR:      LD   HL,PATTAB            ;RUTINA SELECCION CARACTER
3200            LD   (CHSP),HL            ;DIRECCION CARACTER O TABLA SPRITE
3210 CHAR1:     CALL GETNUM               ;RUTINA ENTRADA NUMEROS
3220            LD   (CHARF),A            ;GUARDA NO. CARACTER
3230            CALL TCHAR                ;LO TOMA PARA DEFINIR MATRIZ
3240 YOBBO:     LD   A,0
3250            CALL GTTRIG
3260            CP   0
3270            JR   NZ,YOBBO
3280            RET
3290 GETNUM:    CALL SPPNT                ;IMPRIME ESPACIOS
3300 GN1:       LD   HL,NAMTAB+519
3310            CALL NUMLOP
3320            INC  HL
3330            LD   B,A
3340            CALL NUMLOP
3350            SLA  B
3360            SLA  B
3370            SLA  B
3380            SLA  B
3390            ADD  A,B
3400            RET                       ;TOMA NO. HEX. EN A Y RETORNA
3410 NUMLOP:    PUSH HL
3420 NUMLAP:    LD   C,0
3430            LD   A,0
3440            CALL GTTRIG
3450            CP   0
3460            JR   NZ,NUMLAP            ;ESPERA QUE SE LIBERE BARRA ESPACIO
3470 NUMLLP:    POP  HL
3480 SGLOP:     CALL NPNT                 ;IMPRIME DIGITO
3490            PUSH HL
3500            LD   A,0
3510            CALL GTSTCK
3520            CP   0
3530            JR   Z,NUMELP
3540            INC  C
3550            LD   A,C
3560            CP   16
3570            JR   NZ,NUMELP
3580            LD   C,0
3590 NUMELP:    HALT                      ;SI SE PULSA EL CURSOR INCREMENTA Y ROTA
DIGITO
3600            HALT
3610            HALT
3620            HALT
3630            LD   A,0                  ;ESPERA UN MOMENTO
3640            CALL GTTRIG               ;TERMINADO?
3650            CP   255
3660            JR   NZ,NUMLLP            ;SI NO, REPITE
3670            POP  HL                   ;REPONE HL
```

```
3680          LD    A,C          ;DIGITO EN A
3690          RET                ;RETORNA
3700 NPNT:    LD    A,C          ;NO. EN A
3710          ADD   A,48         ;SE CONVIERTE A ASCII
3720          CP    58           ;ES HEX. A-F
3730          JR    C,NPNT2
3740          ADD   A,7          ;CONVIERTE A ASCII
3750 NPNT2:   CALL  WRTVRM       ;IMPRIMELO
3760          RET
3770 TCHAR:   LD    A,(CHARF)    ;ESTA RUTINA TRANSFIERE
3780          LD    L,A          ;LA DEFINICION DE UN CARACTER O SPRITE
3790          LD    H,0          ;DE LA VRAM A LA MATRIZ DE DEFINICION
3800          ADD   HL,HL
3810          ADD   HL,HL
3820          ADD   HL,HL
3830          PUSH  HL
3840          POP   DE
3850          LD    IX,(CHSP)
3860          ADD   IX,DE
3870          PUSH  IX
3880          POP   HL
3890          LD    C,8
3900          LD    IY,HARRY1
3910 TCOLP:   LD    B,8
3920          CALL  RDVRM
3930 TCILP:   LD    D,0
3940          RLC   A            ;ELEVA BITS DE DEFINICION
3950          JR    NC,TCSKP     ;A BANDERA C
3960          LD    D,1          ;Y ESCRIBE 0 Y 1
3970 TCSKP:   LD    (IY+0),D     ;AL CONJUNTO SEGUN SEA NECESARIO
3980          INC   IY           ;INCREMENTA PUNTERO CONJUNTO
3990          DJNZ  TCILP        ;REPITE
4000          INC   HL           ;INCREMENTA PUNTERO VRAM
4010          DEC   C
4020          JR    NZ,TCOLP     ;8 FILAS
4030          RET
4040 CHSP:    DEFW  PATTAB
4050 CHART:   LD    A,(CHARF)    ;ESTA RUTINA ES
4060          LD    L,A
4070          LD    H,0          ;LA INVERSA DE LA ANTERIOR
4080          ADD   HL,HL
4090          ADD   HL,HL
4100          ADD   HL,HL
4110          PUSH  HL
4120          POP   DE
4130          LD    IX,(CHSP)
4140          ADD   IX,DE
4150          PUSH  IX
4160          POP   HL
4170          LD    C,8
4180          LD    IY,HARRY1
4190 CTLO:    XOR   A
4200          LD    D,A
4210          LD    B,8
4220 CTLI:    LD    A,(IY+0)
4230          CP    1
4240          CCF
4250          RL    D
4260          INC   IY
4270          DJNZ  CTLI
4280          LD    A,D
4290          CALL  WRTVRM
4300          INC   HL
4310          DEC   C
4320          JR    NZ,CTLO
4330          RET
4340 CLR:     LD    IX,HARRY1    ;RUTINA BORRADO MATRIZ
4350          LD    B,64         ;CONJUNTO DE 64 BYTES
4360          LD    A,0          ;0 BORRA
4370 CLRLP:   LD    (IX+0),A     ;LA MATRIZ
4380          INC   IX           ;INCREMENTA PUNTERO MATRIZ
4390          DJNZ  CLRLP        ;64 VECES
```



```
4400          RET                ;RETORNA
4410 INVT:    LD    IX,HARRY1    ;IX PUNTERO DE MATRIZ
4420          LD    B,64         ;MATRIZ DE 64 BYTES
4430 IVTLP:   LD    A,(IX+0)     ;TOMA CONTENIDO MATRIZ
4440          INC   A            ;0 SE HACE 1
4450          AND   1            ;1 SE HACE 0
4460          LD    (IX+0),A     ;LO VUELVE A CARGAR
4470          INC   IX           ;INCREMENTA EL PUNTERO
4480          DJNZ  IVTLP        ;64 VECES
4490          RET
4500 POSIT:   CALL  CURPOS       ;POSICION CURSOR
4510          LD    HL,(CHSP)    ;CARACTER O SPRITE?
4520          XOR   A            ;BORRA A Y BANDERA C
4530          LD    DE,PATTAB    ;DIRECCION TABLA PATRONES GENERADORES
4540          SBC   HL,DE        ;SE HACE UN CARACTER?
4550          JR    NZ,SPOSIT    ;NO,VETE A POSICION SPRITE
4560          LD    A,(CYP)
4570          AND   #00F8
4580          SLA   A
4590          SLA   A
4600          LD    B,A
4610          LD    A,(CXP)
4620          SRL   A
4630          SRL   A
4640          SRL   A
4650          ADD   A,B
4660          LD    HL,NAMTAB
4670          LD    E,A
4680          LD    D,0
4690          ADD   HL,DE        ;CALCULA DIRECCION TABLA NOMBRES
4700          LD    A,(CHARF)
4710          CALL  WRTVRM       ;ESCRIBE CARACTER ACTUAL
4720          RET
4730 SPOSIT:  LD    HL,SATTAB    ;RUTINA PARA SITUAR SPRITE
4740          LD    A,(CHARF)    ;NO. SPRITE
4750          AND   31           ;MOD 32
4760          LD    B,A
4770 SPOSTO:  INC   HL
4780          INC   HL
4790          INC   HL
4800          INC   HL
4810          DJNZ  SPOSTO       ;PLANO DE SPRITE A URILIZAR
4820 SPOST1:  LD    A,(CYP)
4830          CALL  WRTVRM
4840          INC   HL
4850          LD    A,(CXP)
4860          CALL  WRTVRM
4870          INC   HL
4880          LD    A,(CHARF)
4890          CALL  WRTVRM
4900          LD    A,(SPCOL)
4910          INC   HL
4920          CALL  WRTVRM       ;ESCRIBE ATRIBUTOS SPRITE
4930          RET
4940 SPCOL:   DEFB  3            ;CONTIENE COLOR SPRITE
4950 CHCOL:   DEFB  #0031,#0031,#0031,#0031,#0031,#0031,#0031,#0031:COLORES DE
CARACTER
4960 CXP:     DEFB  0
4970 CYP:     DEFB  0
4980 CURPOS:  LD    A,16         ;POSICION DE COMIENZO PARA
4990          LD    (CXP),A      ;SITUAR EL CURSOR
5000          LD    (CYP),A
5010 CUP1:    LD    A,0
5020          CALL  GTTRIG       ;SE HA SOLTADO
5030          CP    255          ;LA BARRA DE ESPACIO?
5040          JR    Z,CUP1
5050 CUPOLP:  LD    HL,(CXP)     ;COGE COORDENADAS X,Y CURSOR
5060          LD    A,0
5070          PUSH  HL
5080          CALL  GTSTCK
5090          POP   HL
5100          CP    1
```

```
5110         JR    NZ,CUP2          ;MUEVE EL CURSOR
5120         DEC   H
5130 CUP2:   CP    3
5140         JR    NZ,CUP3
5150         INC   L
5160 CUP3:   CP    5
5170         JR    NZ,CUP4
5180         INC   H
5190 CUP4:   CP    7
5200         JR    NZ,CUP5
5210         DEC   L
5220 CUP5:   LD    A,H
5230         CP    57
5240         JR    C,CUP6
5250         LD    H,0
5260 CUP6:   LD    (CXP),HL
5270         LD    HL,SATTAB
5280         LD    A,(CYP)
5290         CALL  WRTVRM
5300         INC   HL
5310         LD    A,(CXP)
5320         CALL  WRTVRM
5330         INC   HL
5340         INC   HL
5350         LD    A,12
5360         CALL  WRTVRM
5370         LD    A,0
5380         PUSH  HL
5390         CALL  GTTRIG           ;SE HA TERMINADO EL MOVIMIENTO?
5400         POP   HL
5410         HALT
5420         HALT
5430         HALT
5440         HALT
5450         CP    0
5460         JR    Z,CUPOLP         ;SI NO, SE MUEVE MAS
5470         RET
5480 CSPT:   LD    D,20
5490         LD    HL,NAMTAB+547
5500 CSPT0:  LD    A,32
5510         CALL  WRTVRM
5520         INC   HL
5530         DJNZ  CSPT0
5540         LD    HL,(CHSP)
5550         LD    DE,PATTAB        ;ESTA ES OTRA
5560         LD    IX,CMESS         ;RUTINA DE IMPRESION
5570         SBC   HL,DE            ;DE ESPACIOS Y MENSAJES
5580         JR    Z,CSPT1
5590         LD    IX,SMESS
5600 CSPT1:  LD    HL,NAMTAB+547
5610 CSPT2:  LD    A,(IX+0)
5620         CP    "$"
5630         JR    Z,CSPT3
5640         CALL  WRTVRM
5650         INC   HL
5660         INC   IX
5670         JR    CSPT2
5680 CSPT3:  LD    A,(CHARP)
5690         SRL   A
5700         SRL   A
5710         SRL   A
5720         SRL   A
5730         LD    C,A
5740         CALL  NPNT
5750         INC   HL
5760         LD    A,(CHARP)
5770         AND   15
5780         LD    C,A
5790         CALL  NPNT
5800         RET                    ;ESTAS RUTINAS MANTIENEN EL ESTADO
5810 CMESS:  DEFM  "CAR :$"         ;DEL AREA DE PANTALLA
5820 SMESS:  DEFM  "SPRITE:$"
```



```
5830 COLMES: DEFM  "COLMSPRITE:$"
5840 COLMS1: DEFM  "COLMCAR:$"
5850 COLPT:  LD    HL,NAMTAB+579
5860         LD    IX,COLMES
5870 COLPT1: LD    A,(IX+0)
5880         CP    "$"
5890         JR    Z,COLPT2
5900         CALL  WRTVRM
5910         INC   HL
5920         INC   IX
5930         JR    COLPT1
5940 COLPT2: LD    A,(SPCOL)
5950         SRL   A
5960         SRL   A
5970         SRL   A
5980         SRL   A
5990         LD    C,A
6000         CALL  NPNT
6010         LD    A,(SPCOL)
6020         AND   15
6030         LD    C,A
6040         INC   HL
6050         CALL  NPNT
6060         LD    HL,NAMTAB+532
6070         LD    IX,COLMS1
6080 COLPT3: LD    A,(IX+0)
6090         CP    "$"
6100         JR    Z,COLPT4
6110         CALL  WRTVRM
6120         INC   HL
6130         INC   IX
6140         JR    COLPT3
6150 COLPT4: INC   HL
6160         PUSH  HL
6170         LD    A,(CHARP)
6180         LD    L,A
6190         LD    H,0
6200         ADD   HL,HL
6210         ADD   HL,HL
6220         ADD   HL,HL
6230         PUSH  HL
6240         POP   DE
6250         LD    IX,COLTAB
6260         ADD   IX,DE
6270         PUSH  IX
6280         POP   HL
6290         LD    IX,CHCOL
6300         LD    B,8
6310 COLPT5: CALL  RDVRM
6320         LD    (IX+0),A
6330         INC   IX
6340         INC   HL
6350         DJNZ  COLPT5
6360         POP   HL
6370         LD    B,8
6380         LD    DE,31
6390         LD    IX,CHCOL
6400 COLPT6: LD    A,(IX+0)
6410         PUSH  AF
6420         SRL   A
6430         SRL   A
6440         SRL   A
6450         SRL   A
6460         LD    C,A
6470         CALL  NPNT
6480         POP   AF
6490         AND   15
6500         INC   HL
6510         LD    C,A
6520         CALL  NPNT
6530         ADD   HL,DE
6540         INC   IX
```

```
6550          DJNZ COLPT6
6560          RET
6570 BANK:    DEFB 0
6580 BPMESS:  DEFM "BANCO:$"
6590 BANKPT:  LD   HL,NAMTAB+611
6600          LD   IX,BPMESS
6610 BKPT1:   LD   A,(IX+0)
6620          CP   "$"
6630          JR   Z,BKPT2
6640          CALL WRTVRM
6650          INC  HL
6660          INC  IX
6670          JR   BKPT1
6680 BKPT2:   LD   A,(BANK)
6690          AND  15
6700          LD   C,A
6710          CALL NPNT
6720          RET
6730 FTTXRM:  LD   HL,PATTAB
6740          LD   DE,#B000
6750          LD   BC,#1800
6760          CALL LDIRMV
6770          LD   HL,COLTAB
6780          LD   DE,#C800
6790          LD   BC,#1800
6800          CALL LDIRMV
6810 SPTXRM:  LD   HL,SPTTAB
6820          LD   DE,#E000
6830          LD   BC,#0800
6840          CALL LDIRMV
6850          RET
6860 LPTXRM:  LD   A,(BANK)
6870          LD   H,A
6880          LD   L,0
6890          ADD  HL,HL
6900          ADD  HL,HL
6910          ADD  HL,HL
6920          PUSH HL
6930          LD   DE,#B000
6940          ADD  HL,DE
6950          EX   DE,HL
6960          LD   HL,PATTAB
6970          LD   BC,#0800
6980          CALL LDIRMV
6990          POP  HL
7000          LD   DE,#C800
7010          ADD  HL,DE
7020          EX   DE,HL
7030          LD   HL,COLTAB
7040          LD   BC,#0800
7050          CALL LDIRMV
7060          JR   SPTXRM
7070 COLR:    CALL SPPNT
7080          LD   A,0
7090          CALL GTTRIG
7100          CP   255
7110          JR   Z,COLR
7120          LD   HL,(CHSP)
7130          XOR  A
7140          LD   DE,PATTAB
7150          SBC  HL,DE
7160          JR   NZ,COLSP
7170          LD   IX,CHCOL
7180          LD   B,8
7190 COLLYP:  LD   A,8
7200          SUB  B
7210          ADD  A,48
7220          PUSH BC
7230          LD   HL,NAMTAB+517
7240          CALL WRTVRM
7250          INC  HL
7260          LD   A,":"
```



```
7270          CALL WRTVRM
7280          INC  HL
7290          LD   A,(IX+0)
7300          SRL  A
7310          SRL  A
7320          SRL  A
7330          SRL  A
7340          LD   C,A
7350 HNGON1:  LD   A,0
7360          CALL GTTRIG
7370          CP   255
7380          JR   Z,HNGON1
7390          PUSH IX
7400          CALL SGLOP
7410          POP  IX
7420          SLA  A
7430          SLA  A
7440          SLA  A
7450          SLA  A
7460          LD   B,A
7470          LD   A,(IX+0)
7480          AND  15
7490          OR   B
7500          LD   (IX+0),A
7510          AND  15
7520          LD   C,A
7530          INC  HL
7540 HANGON:  LD   A,0
7550          CALL GTTRIG
7560          CP   255
7570          JR   Z,HANGON
7580          PUSH IX
7590          CALL SGLOP
7600          POP  IX
7610          AND  15
7620          LD   B,A
7630          LD   A,(IX+0)
7640          AND  #00F0
7650          OR   B
7660          LD   (IX+0),A
7670          INC  IX
7680          POP  BC
7690          DJNZ COLLYP
7700          LD   A,(CHARP)
7710          LD   L,A
7720          LD   H,0
7730          ADD  HL,HL
7740          ADD  HL,HL
7750          ADD  HL,HL
7770          LD   DE,COLTAB
7780          ADD  HL,DE
7790          EX   DE,HL
7800          LD   HL,CHCOL
7810          LD   BC,8
7820          CALL LDIRVM
7830          RET
7840 COLSP:   CALL GETNUM
7850          LD   (SPCOL),A
7860          RET
7870 MSP:     LD   A,(#9001)        ; COGE VALOR ACTUAL VDP(1)
7880          OR   1                ;BANDERA AMPLIACION 1
7890          LD   (#9001),A        ;LA DEVUELVE
7900          LD   B,A
7910          LD   C,1
7920          CALL WRTVRM           ;SE ESCRIBE EN VDP
7930          RET
7940 DMSP:    LD   A,(#9001)
7950          AND  #00FE            ;BANDERA AMPLIACION 0
7960          LD   (#9001),A
7970          LD   B,A
7980          LD   C,1
7990          CALL WRTVDP           ;SE ESCRIBE EN VDP
```

```
8000              RET
8010 SP8:         LD    A,(#9001)
8020              AND   #00FD                 ;BANDERA SPRITE 16*16 0
8030              LD    (#9001),A
8040              LD    B,A
8050              LD    C,1
8060              CALL  WRTVDP                ;SE ESCRIBE EN VDP
8070              RET
8080 SP16:        LD    A,(#9001)             ;BANDERA SPRITE 16*16 1
8090              LD    (#9001),A
8100              LD    B,A
8110              LD    C,1
8120              CALL  WRTVDP                ;SE ESCRIBE EN VDP
8130              RET
8140 LSCRL:       LD    IX,HARRY1             ;DESPLAZAMIENTO IZQUIERDA DESDE CONJUNTO
1 A 2
8150              LD    IY,HARRY2
8160              LD    C,8
8170 LSLP:        LD    A,(IX+0)
8180              LD    (IY+7),A
8190              LD    B,7
8200              INC   IX
8210 LSLP1:       LD    A,(IX+0)
8220              LD    (IY+0),A
8230              INC   IX
8240              INC   IY
8250              DJNZ  LSLP1
8260              INC   IY
8270              DEC   C
8280              JR    NZ,LSLP
8290              LD    IX,HARRY1             ;PONE CONJUNTO 2 EN 1
8300              LD    IY,HARRY2
8310              LD    B,64
8320 LSLP2:       LD    A,(IY+0)
8330              INC   IX
8340              INC   IY
8350              DJNZ  LSLP2                 ;64 BYTES
8360 LSLP3:       LD    A,0
8370              CALL  GTTRIG                ;ESPERA QUE SE LIBERE BARRA ESPACIO
8380              CP    255
8390              JR    Z,LSLP3
8400              RET
8410 RSCRL:       LD    B,7                   ;SCROLL DERECHO
8420 RSCRLP:      PUSH  BC                    ;SCROLL 7 VECES
8430              CALL  LSCRL
8440              POP   BC
8450              DJNZ  RSCRLP
8460              RET
8470 USCRL:       LD    DE,8                  ;SCROLL ARRIBA SIMILAR A IZQUIERDO
8480              LD    IX,HARRY1
8490              LD    IY,HARRY2
8500              LD    C,8
8510 USRLP1:      PUSH  IX
8520              PUSH  IY
8530              LD    A,(IX+0)
8540              LD    (IY+56),A
8550              ADD   IX,DE
8560              LD    B,7
8570 USRLP2:      LD    A,(IX+0)
8580              LD    (IY+0),A
8590              ADD   IX,DE
8600              ADD   IY,DE
8610              DJNZ  USRLP2
8620              POP   IY
8630              POP   IX
8640              INC   IY
8650              INC   IX
8660              DEC   C
8670              JR    NZ,USRLP1
8680              LD    IX,HARRY1
8690              LD    IY,HARRY2
8700              LD    B,64
```



```
8710 USRLP3:      LD    A,(IY+0)
8720              LD    (IX+0),A
8730              INC   IX
8740              INC   IY
8750              DJNZ  USRLP3
8760 USRLP4:      LD    A,0
8770              CALL  GTTRIG
8780              CP    255
8790              JR    Z,USRLP4
8800              RET
8810 DSCRL:       LD    B,7                   ;SCROLL HACIA ABAJO
8820 DCRLP:       PUSH  BC
8830              CALL  USCRL
8840              POP   BC
8850              DJNZ  DCRLP
8860              RET
8870 FVER:        LD    IX,HARRY1             ;ESPEJO VERTICAL CONJUNTO 1
8880              LD    IY,HARRY2+56          ;CONJUNTO 2
8890              LD    C,8
8900              LD    D,8
8910 FVER1:       PUSH  IX
8920              PUSH  IY
8930              LD    B,8
8940 FVER2:       LD    A,(IX+0)
8950              LD    (IY+0),A
8960              INC   IX
8970              INC   IY
8980              DJNZ  FVER2
8990              POP   HL
9000              XOR   A
9010              SBC   HL,DE
9020              PUSH  HL
9030              POP   IY
9040              POP   IX
9050              ADD   IX,DE
9060              DEC   C
9070              JR    NZ,FVER1
9080              LD    B,64
9090              LD    IX,HARRY1             ;PONE CONJUNTO TRANSFORMADO 2 EN 1
9100              LD    IY,HARRY2
9110 FVER3:       LD    A,(IY+0)
9120              LD    (IX+0),A
9130              INC   IX
9140              INC   IY
9150              DJNZ  FVER3
9160              JP    USRLP4
9170 FHOR:        LD    IX,HARRY1             ;ESPEJO HORIZONTAL SIMILAR VERTICAL
9180              LD    IY,HARRY2+7
9190              LD    C,8
9200 FHOR1:       PUSH  IY
9210              LD    DE,8
9220              LD    B,8
9230 FHOR2:       LD    A,(IX+0)
9240              LD    (IY+0),A
9250              INC   IX
9260              DEC   IY
9270              DJNZ  FHOR2
9280              POP   IY
9290              XOR   A
9300              ADD   IY,DE
9310              DEC   C
9320              JR    NZ,FHOR1
9330              LD    B,64
9340              LD    IX,HARRY1
9350              LD    IY,HARRY2
9360 FHOR3:       LD    A,(IY+0)
9370              LD    (IX+0),A
9380              INC   IX
9390              INC   IY
9400              DJNZ  FHOR3
9410              JP    USRLP4
9420 BANKSW:      CALL  SPPNT
```

```
9430            LD    HL,NAMTAB+157
9440            CALL  NUMLOP
9450            CP    3
9460            JR    NC,BANKSW
9470  BSW1:     LD    (BANK),A
9480            LD    BC,#0800
9490            LD    DE,#B000
9500            LD    H,A
9510            LD    L,0
9520            ADD   HL,HL
9530            ADD   HL,HL
9540            ADD   HL,HL
9550            PUSH  HL
9560            ADD   HL,DE
9570            LD    DE,PATTAB
9580            CALL  LDIRVM
9590            POP   HL
9600            LD    DE,#C800
9610            ADD   HL,DE
9620            LD    DE,COLTAB
9630            LD    BC,#0800
9640            CALL  LDIRVM
9650            CALL  TCHAR
9660            RET
9670  COPBNK:   DEFB  0
9680  COPCHA:   DEFB  0
9690  COPY:     CALL  SPPNT                 ;RUTINA COPIA DEFINICION
9700            LD    IX,BPMESS
9710            LD    HL,NAMTAB+517
9720  COPYLP:   LD    A,(IX+0)
9730            CP    "$"
9740            JR    Z,COPY1
9750            CALL  WRTVRM
9760            INC   HL
9770            INC   IX
9780            JR    COPYLP                ;IMPRIME MENSAJE
9790  COPY1:    CALL  NUMLOP                ;COGE NO. BANCO
9800            LD    (COPBNK),A            ;GUARDA BANCO A COPIAR
9810            CALL  GETNUM                ;COGE NO. CARACTER/SPRITE
9820            LD    (COPCHA),A
9830            LD    A,(COPBNK)
9840            CP    4
9850            JR    NC,COPY
9860            CP    3
9870            JP    Z,SPCOPY              ;COPIA CARACTER O SPRITE
9880            LD    A,(COPCHA)
9890            LD    L,A
9900            LD    A,(COPBNK)
9910            LD    H,A
9920            ADD   HL,HL
9930            ADD   HL,HL
9940            ADD   HL,HL                 ;CALCULA DIRECCION CARACTER
9950            PUSH  HL
9960            LD    DE,#B000
9970            ADD   HL,DE                 ;EN RAM
9980            PUSH  HL
9990            LD    A,(CHARF)
10000           LD    L,A
10010           LD    H,0
10020           ADD   HL,HL
10030           ADD   HL,HL
10040           ADD   HL,HL
10050           LD    DE,(CHSP)
10060           ADD   HL,DE
10070           EX    DE,HL                 ;COGE DIRECCION CARACTER/SPRITE DE VF
10080           POP   HL
10090           LD    BC,8
10100           CALL  LDIRVM                ;LA TRANSFIERE DE RAM A VRAM
10110           POP   BC
10120           LD    DE,(CHSP)
10130           LD    HL,PATTAB
10140           XOR   A
```



```
10150           SBC   HL,DE
10160           JR    NZ,COPSKP
10170           PUSH  BC
10180           POP   HL
10190           LD    DE,#C800              ;SI SE TRANSFIERE UN CARACTER
10200           ADD   HL,DE
10210           PUSH  HL
10220           LD    DE,COLTAB
10230           LD    A,(CHARF)
10240           LD    L,A
10250           LD    H,0
10260           ADD   HL,HL
10270           ADD   HL,HL
10280           ADD   HL,HL
10290           ADD   HL,HL
10300           EX    DE,HL
10310           POP   HL
10320           LD    BC,8
10330           CALL  LDIRVM                ;TOMA ELEMENTO TABLA COLOR
10340 COPSKP:   CALL  TCHAR                 ;PONE CARACTER EN MATRIZ
10350           RET
10360 SPCOPY:   LD    A,(COPCHA)            ;RUTINA COPIA SPRITE
10370           LD    L,A
10380           LD    H,0
10390           ADD   HL,HL
10400           ADD   HL,HL
10410           ADD   HL,HL
10420           LD    DE,#E000
10430           ADD   HL,DE
10440           PUSH  HL
10450           LD    A,(CHARF)
10460           LD    L,A
10470           LD    H,0
10480           ADD   HL,HL
10490           ADD   HL,HL
10500           ADD   HL,HL
10510           LD    DE,(CHSP)
10520           ADD   HL,DE
10530           EX    DE,HL
10540           POP   HL
10550           LD    BC,8
10560           CALL  LDIRVM
10570           CALL  TCHAR
10580           RET
10590 LOAD:     LD    A,1                   ;CARGAR ES LA FUNCION BASIC 1
10600 L1:       LD    (BASFNC),A
10610           POP   HL                    ;COGE DIRECCION RETORNO DE LA PILA
10620           LD    HL,L2                 ;COGE NUEVA DIRECCION RETORNO
10630           LD    (RTADDR),HL           ;LA GUARDA PARA BASIC
10640           RET                         ;A BASIC
10650 L2:       LD    A,(BANK)
10660           CALL  GTX
10670           CALL  TCHAR
10680           JP    NDFNC
10690 SAVE:     LD    A,2                   ;GUARDAR ES LA FUNCION BASIC 2
10700           JR    L1                    ;VA A RUTINA DE CARGA
10710 GTX:      LD    HL,#B000
10720           LD    DE,PATTAB
10730           LD    BC,#1800
10740           CALL  LDIRVM
10750           LD    HL,#C800
10760           LD    DE,COLTAB
10770           LD    BC,#1800
10780           CALL  LDIRVM
10790           LD    HL,#E000
10800           LD    DE,SPTTAB
10810           LD    BC,#0800
10820           CALL  LDIRVM
10830           RET
10840 BASFNC:   DEFB  0                     ;GUARDA NO. FUNCION
10850 RTADDR:   DEFW  0                     ;Y DIRECCION RETORNO PARA BASIC
10860           END
```

El programa anterior se ensamblará y guardará en cassette con el nombre "GM2". El siguiente programa en BASIC será, asímismo, escrito y almacenado. Este programa cargará y ejecutará el código máquina.

```
10 CLEAR 200,&H8FFF
20 PRINT "CARGANDO..."
30 BLOAD "CAS:GM2"
40 DEFUSR=&H9008
50 A=USR(A)
60 DEFUSR=PEEK(&H9AED)+256*(PEEK(&H9AEE)):REM DIRECCION RETO
RNO
70 IF PEEK(&H9AEC)=0 THEN SCREEN 1,0,0,1:END
80 IF PEEK(&H9AEC)=1 THEN BLOAD "CAS:CHARS"
90 IF PEEK(&H9AEC)=2 THEN BSAVE "CAS:CHARS",&HB000,&HE801
100 GOTO 50
```

UTILIZACION DEL PROGRAMA EN MODO GRAFICO II

Al ejecutar el programa, la pantalla quedará dividida en tres zonas. La zona superior en negro, la zona intermedia conteniendo una retícula de 10 por 8 (las columnas de la izquierda y derecha están marcadas con una línea de cruces), y la tercera zona que visualiza las funciones ejecutadas por el programa.

Moviendo el cursor sobre las columnas izquierda y derecha de la segunda zona, aparecerán mensajes bajo la retícula, en la tercera zona de la pantalla. El cursor se mueve mediante sus propias teclas, y la barra de espacio tiene dos funciones: cuando el cursor está en el centro de la matriz de definición, si se pulsa la barra de espacio, el cursor alternará su estado en la matriz de activado a desactivado, y viceversa. Cuando el cursor se sitúa en una de las columnas de función, la pulsación de la barra de espacio ejecutará la función mostrada bajo la matriz de carácter.

Además de los mensajes de función, la tercera zona de la pantalla mostrará ciertos mensajes de estado, la mayoría de los cuales se explican por sí mismos.

Las funciones ejecutadas por el programa, son las siguientes:

SALIDA	Finaliza el programa.
CARGA	Carga un archivo llamado "CHARS" desde el cassette, conteniendo las definiciones de sprites y carácteres y los colores de carácteres
GUARDA	Almacena el archivo "CHARS" en cassette.
BORRA	Borra la matriz de definición.
CARACTER	Selecciona el carácter a modificar, y transfiere su definición de la VRAM a la matriz de definición.
SPRITE	Selecciona el sprite a definir.
COLOR	Permite definir los colores de un carácter, fila a fila. Al definir un sprite, fija el color de la función POSICION.

SCROLL IZQ DER ARR ABA	Realiza un scroll rotacional
COPIA	Copia un carácter o definición de sprite, al carácter o definición actual.
INVIERTE	Invierte todos los puntos de la definición existente.
SIMET HOR VER	Construye una definición simétrica con relación a un eje horizontal o vertical, trazado por el centro de la definición.
MENU 1/2	Alterna los menús de función accesibles mediante las columnas de función.
POSICION	Permite situar los sprites o carácteres en la parte superior de la pantalla.
BANCO	Como hemos visto, la tabla de patrones en el modo gráfico II, consiste en tres juegos de 256 definiciones de carácter. El número de banco hace referencia a cada uno de estos tres juegos de definiciones, al definir un carácter, y los de aquéllos que aparecen en el tercio de la pantalla. (Cuando la función COPY solicita un número del banco, los números 0 á 2, se refieren a los tres bancos de carácteres, y el número 3 a las definiciones de sprite).
AMPLIACION REDUCCION SPRITES	Estas dos funciones modifican los registros del VDP para la ampliación o reducción de sprites.
SPRITES 8*8/16*16	

La entrada numérica se considera formada por dos dígitos hexadecimales. Estos dígitos se seleccionan mediante las teclas de cursor, orden superior seguido por orden inferior.

A la salida del programa, los datos de carácteres permanecen en la RAM de la CPU en las siguientes posiciones:

B000H a C800H	La tabla de patrones generadores (es decir, las definiciones de carácteres).
C800H a E000H	La tabla de color.
E000H a E800H	Las definiciones de sprite.

UTILIZACION DEL PROGRAMA EN OTROS MODOS

En el modo gráfico I hay 256 definiciones de carácter posibles, y por lo tanto, sólo se precisa definir el primer banco. Estas definiciones se encuentran entre B000H y B800H (la tabla de color puede ignorarse) y las definiciones de sprite permanecen activas.

Para utilizar el programa en el modo de texto, sigue vigente lo anteriormente expuesto, salvo que los sprites no estén disponibles, y en consecuencia, sus definiciones carecen de importancia. Debe recordarse que en el modo de texto solamente se utilizan las seis columnas de la izquierda, siendo aconsejable dejar en blanco las dos columnas de la derecha.

DEFINICION DINAMICA DE PATRONES

Como el VDP mantiene todas sus definiciones de caracteres en la VRAM, es posible conseguir efectos especiales en el plano patrón, redefiniendo un carácter mientras se ejecuta un programa. La nueva definición aparecerá en pantalla en todas aquellas posiciones en las que se haga referencia a dicha definición en la tabla de nombres. De esta forma, grandes zonas de la pantalla pueden cambiarse alterando un número relativamente pequeño de elementos de la tabla de patrones generadores. Como ejemplo, consideraremos la visualización de una escalera de 7 caracteres de alto y uno de ancho, que aparecerá como un ascensor con sus peldaños ascendiendo por la pantalla, y que se definirá utilizando un único carácter.

En primer lugar, crearemos la definición del carácter básico. Se necesita un carácter que aparezca como parte de una escalera, y que se enlaza consigo mismo verticalmente. Utilizaremos la siguiente definición de patrón:

```
10 ;10000001   #81
20 ;11111111   #FF
30 ;10000001   #81
40 ;10000001   #81
50 ;10000001   #81
60 ;11111111   #FF
70 ;10000001   #81
80 ;10000001   #81
```

Habiendo decidido la apariencia de nuestro carácter básico, debe insertarse en la VRAM; lo situaremos en el lugar del carácter 0.

```
 10 CHRDEF:  DEFB #81,#FF,#81,#81,#81,#FF,#81,#81
 20 CHARIN:  LD   B,8          ;NO. ELEMENTOS CHRDEF
 30          LD   IX,CHRDEF    ;DIR. DEFINICION IX
 40          LD   HL,0000      ;DIRECCION PGT EN HL
 50 CILP:    LD   A,(IX+0)     ;BYTE DE DATOS
 60          CALL #004D        ;ESCRIBE (HL) EN VRAM
 70          INC  HL
 80          INC  IX           ;INCREMENTA PUNTEROS
 90          DJNZ CILP         ;OCHO VECES
100          RET
```

Tenemos, así, una definición en la VRAM, que podemos alterar para conseguir el efecto deseado. La manipulación requerida consiste en un scroll vertical de los 8 bytes de la definición del carácter, haciendo que el byte superior se convierta en el inferior. Para ello se empleará la rutina siguiente:

```
 110 RDVRM:   EQU  #004A
 120 WRTVRM:  EQU  #004D
 130 DSCROL:  LD   B,7          ;PREPARA EL CONTADOR
 140          LD   HL,0000      ;PREPARA LA DIRECCION
 DE DEFINICION
 150          CALL RDVRM        ;TOMA EL PRIMER VALOR
 160          LD   C,A          ;LO GUARDA PARA DESPU
ES
 170 DSCLP:   INC  HL           ;INCREMENTA EL PUNTER
O VRAM
 180          CALL RDVRM        ;TOMA DEFINICION DE L
INEA
```



```
 190          DEC  HL           ;Y ALMACENA ...
 200          CALL WRTVRM       ;UNA LINEA DE DEFINIC
ION
 210          INC  HL           ;REPOSICIONA PUNTERO
VRAM
 220          DJNZ DSCLP        ;SIETE VECES
 230          LD   A,C          ;LINEA SUPERIOR ANTER
IOR
 240          CALL WRTVRM       ;SE CONVIERTE EN INFE
RIOR
 250          RET
```

Llamando a la rutina superior en forma repetida, nuestro carácter escalera aparecerá con peldaños ascendentes. Pueden construirse escaleras de cualquier tamaño. También se necesitará el siguiente programa:

```
260 START:   CALL CHARIN       ;DEFINE CARACTER
270          LD   HL,6144+8    ;TABLA NOMBRES +8
280          LD   DE,32        ;LONGITUD LINEA EN DE
290          LD   B,7          ;ESCALERA 7 CARACTS.
300          LD   A,0          ;CARACTER ESCALERA=0
310 PLOOP:   CALL WRTVRM       ;IMPRIME ESCALERA
320          ADD  HL,DE        ;IMPRIME MAS ABAJO
330          DJNZ PLOOP        ;LOS SIETE CARACTERES
340 LOOP2:   CALL DSCROL       ;DESPLAZA DEFINICION
350          HALT
360          HALT              ;ESPERA 2/50 SEG.
370          JR   LOOP2        ;REPITE
```

Como puede observarse en este ejemplo, el principio de la definición dinámica de carácteres es una herramienta muy potente, estando únicamente limitado por la imaginación. En realidad, es este principio el que nos permitirá considerar el modo gráfico II como un modo reflejado, bit a bit, en la memoria.

MODO GRAFICO II COMO UN SISTEMA DE MAPA DE BITS

Si en el modo grafico II, disponemos la tabla de nombres de forma que los 768 bytes de la tabla estén numerados consecutivamente de 0 a 255, tres veces, cada espacio de un carácter tendrá su propia definición en la tabla de patrones generadores, y por tanto, podrá considerarse este modo como un sistema de mapa de bits. Véase la figura 6.1. Si disponemos la tabla de nombres de esta manera, tendremos, en efecto, un modo de mapa de bits con 256*192 puntos en alta resolución, con 15 colores disponibles (aunque, desafortunadamente, la resolución del color horizontal está limitada a 32 bloques de 8 puntos).

Si consideramos que el origen sea la esquina inferior izquierda de la pantalla, la forma para calcular qué byte de la tabla de patrones generadores contiene el punto al que hace referencia la coordenada X,Y es la siguiente:

(Dirección de la base de la tabla de patrones)+((((191-Y)div 8)*256)+((191-Y)mod 8)+(X AND OF8H))

y el bit, que hay que poner a nivel 1, dentro de dicho byte, se obtiene mediante:

7-(X AND 7)

Utilizando las fórmulas anteriores, la siguiente rutina iluminará el punto especificado por las coordenadas HL (L contiene la coordenada X y H la coordenada Y) del color fijado por el acumulador.

```
   10 SETXY:       PUSH AF             ;GUARDA CODIGO DE COL
OR
   20              LD   A,191          ;HACE Y=...
   30              SUB  H              ;=191-Y
   40              LD   H,A            ;LO CARGA EN H
   50              PUSH HL             ;GUARDA X,Y
   60              SRL  H
   70              SRL  H
   80              SRL  H              ;Y=Y/8
   90              LD   D,H            ;EN PARTE ALTA DE=*25
6
  100              POP  HL             ;RESTAURA X,Y
  110              LD   A,H            ;CONSIGUE Y
  120              AND  7              ;MOD 8
  130              LD   E,A            ;EN PARTE BAJA DE
  140              XOR  A              ;BANDERA C=0
  150              LD   A,L            ;CONSIGUE X
  160              AND  #00F8          ;
  170              LD   E,A            ;SE SUMA A DE
  180              LD   A,0            ;
  190              ADC  A,D            ;TERMINA SUMA 16 BIT
  200              LD   D,A            ;DE CONTIENE EL DESPL
AZAMIENTO DEL GENERADOR DE PATRONES Y TABLA DE COLO
  210              LD   B,L            ;GUARDA X
  220              LD   HL,8192        ;DIRECCION TABLA COLO
R
  230              ADD  HL,DE          ;SUMA DESPLAZAMIENTO
  240              CALL #004A          ;CONSIGUE COLOR
  250              AND  15             ;ENMASCARA COLOR DE F
ONDO
  260              LD   C,A            ;SE GUARDA EN C
  270              POP  AF             ;RESTAURA COLOR
  280              SLA  A              ;ELEVA CODIGO DE COLO
R
  290              SLA  A
  300              SLA  A              ;AL NIBBLE SUPERIOR
  310              SLA  A
  320              ADD  A,C            ;SUMA COLOR DE FONDO
  330              CALL #004D          ;Y LO DEVUELVE
  340              LD   A,7
  350              AND  B
  360              LD   B,A
  370              XOR  A
  380              INC  B
  390              CCF
  400 MLP:         RRA                 ;PONE BANDERA C EN A
 PARA FORMAR
  410              DJNZ MLP            ;MASCARA
  420              LD   HL,0000        ;DIRECCION DEL GENERA
DOR DE PATRONES EN HL
  430              ADD  HL,DE          ;SUMA DESPLAZAMIENTO
  440              CALL #004A          ;CONSIGUE VALOR ANTER
IOR
  450              XOR  B              ;LO ENMASCARA CON B
  460              CALL #004D          ;LO DEVUELVE
  470              RET                 ;FIN
```

Si consideramos los fundamentos del modo gráfico II, puede verse que es sencillo disponer varios mapas de memoria diferentes, manipulando los elementos de la tabla de nombres. Sin embargo, en la práctica el mapa de memoria descrito anteriormente, y en la figura 6.7, es tan fácil de usar como el que más, y en muchos casos, más apropiado.

Debe recordarse, también, que aunque la técnica anterior nos permite tratar el modo gráfico II como un mapa de bits, todavía podemos realizar operaciones como si fuera del tipo de mapa de carácteres. Así, la pantalla completa puede desplazarse verticalmente (scroll) en saltos de 8 puntos, manipulando unicamente 768 bytes de VRAM. Alternativamente podemos definir el último bloque de definiciones de caracteres (aquéllos que se corresponden con las referencias del tercio inferior de la tabla de nombres) como un juego standard, y tratar los dos tercios superiores de la pantalla, como si fueran de mapa de bits. Sin duda, el lector descubrirá otras muchas posibilidades

TECNICAS DE INTERRUPCION CON SPRITES

Hay dos problemas fundamentales en el manejo de sprites con el VDP TMS-9929 A. En primer lugar, los sprites son de un único color cada uno, y en segundo, todos los sprites deben ser del mismo tamaño y ampliación, simultáneamente. Sin embargo, es posible crear sprites que sean, aparentemente, de dos colores, y también es posible visualizar sprites de dos tamaños o ampliaciones diferentes. Para ello, debe conmutarse, mediante interrupciones, entre las tablas de sprite. Cuando se accede al VDP durante una interrupción es esencial que la interrupción no se produzca durante el tiempo de acceso del programa de usuario al VDP. Esto puede sonseguirse, bien situando todos los accesos del programa al VDP detrás de una instrucción HALT del Z-80, o bien, señalizando mediante banderas que se está produciendo un acceso al VDP.

SPRITES DE DOS COLORES

Si alternamos los elementos del nombre de sprite y del color de sprite en la tabla de atributos de sprite, después de cada interrupción de barrido de cuadro, es posible producir sprites que aparezcan en dos colores diferentes. Para ello, sugerimos el siguiente procedimiento:

En primer lugar, crearemos dos definiciones de sprite, una por cada color. Las definiciones se situarán en la VRAM consecutivamente, de forma que las definiciones 0 y 1 pertenezcan a un sprite multicolor, las definiciones 2 y 3 a otro, y así, sucesivamente.

En segundo lugar, reservaremos memoria para una tabla de conmutación de sprites. Cada elemento de esta tabla contendrá un byte con el código de color para la definición de sprite par (en su nibble de orden superior) y el código de color para la definición de sprite impar (en su nibble de orden inferior). Esta tabla deberá escribirse siempre que se coloque un nuevo sprite en la tabla de atributos de sprite.

Por último, debe prepararse una rutina, controlada por las interrupciones, que conmute las dos definiciones y los dos colores en cada inte-

rrupción de barrido de cuadro. La siguiente rutina desarrolla este proceso:

```
   10 SPSWTB:   DEFS 32          ;TABLA CONMUTACION COLOR
   20 SPSWIT:   LD   HL,6912     ;DIRECCION BASE TABLA ATRIBUTOS SPRITE EN HL
   30           LD   DE,SPSWTB   ;DIRECCION TABLA CONMUTACION COLOR EN DE
   40           LD   B,32        ;NO. MAX. ELEMENTOS
   50 SPSWLP:   CALL #004A       ;LEE POSICION VERTICAL SPRITE
   60           CP   #00D0       ;ES MARCADOR FIN SPRITE?
   70           RET  Z           ;SI, VUELVE
   80           INC  HL          ;NO, INCREMENTA PUNTERO
   90           INC  HL          ;DOS VECES
  100           CALL #004A       ;LEE NO. PATRON
  110           XOR  1           ;PATRON + 0 -1
  120           LD   C,A         ;GUARDA NUEVO NO. PATRON
  130           CALL #004D       ;LO ESCRIBE EN ATRIBUTOS SPRITE
  140           INC  HL          ;APUNTA ROTULO SPRITE
  150           LD   A,(DE)      ;COGE COLOR SPRITE
  160           AND  A           ;BORRA BANDERA EXCESO
  170           RR   C           ;ROTA BIT 0 NO. PATRON EN EXCESO
  180           JR   C,SKP       ;SI PATRON PAR, NO ...
  190           SLA  A           ;ELEVA NIBBLE SUPERIOR DE COLOR
  200           SLA  A
  210           SLA  A
  220           SLA  A           ;EN NIBBLE INFERIOR
  230 SKP:      AND  15          ;ENMASCARA NIBBLE SUPERIOR
  240           LD   C,A         ;GUARDA CODIGO COLOR EN C
  250           CALL #004A       ;LEE ROTULO ANTERIOR
  260           AND  #0080       ;ENMASCARA BIT PREVIO DE RELOJ
  270           OR   C           ;SUMA NUEVO COLOR
  280           CALL #004D       ;ESCRIBE NUEVO ROTULO EN ATRIBUTOS SPRITE
  290           INC  HL          ;APUNTA SIGUIENTE ELEMENTO TABLA ATRIBUTOS SPRITE
  300           INC  DE          ;APUNTA SIGUIENTE ELEMENTO TABLA CONMUTACION
  310           DJNZ SPSWLP      ;32 VECES
  320           RET              ;VUELVE A INTERRUPCION S.O.
```


Así, dos sprites de distintos colores y definiciones, comparten el mismo elemento de la tabla de atributos de sprite, visualizándose cada uno alternativamente en cada cuadro de la pantalla de televisión. Debido a esto, se apreciará una ligera vibración, si bien en circunstancias normales, la persistencia del tubo de pantalla será suficiente para que esta vibración sea casi insignificante.

SPRITES DE DIFERENTES TAMAÑOS: CONMUTACION DE LOS REGISTROS DEL VDP MEDIANTE INTERRUPCIONES

Para visualizar sprites de diferentes tamaños o amplitudes en pantalla simultáneamente, se requieren dos tablas de atributos de sprite, que deben ubicarse en la VRAM al mismo tiempo (una para cada tamaño o amplitud). Después, se debe hacer que el VDP acceda a una de ellas, y a continuación, a la otra, en cada cuadro.

En el modo multicolor y en el modo gráfico I, se puede, normalmente, diseñar un mapa de la memoria VRAM que permita contener dichas tablas sin superponerlas con otros subbloques del VDP. En el modo gráfico II, sin embargo, se hace necesario superponer la segunda tabla de atributos de sprite, sobre una de las otras tablas. El procedimiento más satisfactorio consiste en robar las últimas 16 definiciones de sprite para este propósito, tal y como se muestra en el siguiente mapa de memoria:

0000H	Tabla de generación de carácteres
1800H	Tabla de nombres de patrón
1B00H	Tabla de atributos de sprite número 1
2000H	Tabla de color de carácteres
3800H	Tabla de patrones generadores de sprite
3F80H	Tabla de atributos de sprite número 2

Habiendo diseñado un mapa de memoria adecuado, es muy simple escribir una rutina de interrupción, que conmute los bits de tamaño y/o amplitud en el registro 1, y la dirección base de la tabla de atributos de sprite en el registro 5. La siguiente rutina nos muestra este procedimiento, manteniendo dos tablas de atributos, una para sprites de 16*16 sin aumentar, y otra para sprites de 16*16 aumentados.

```
   10 VAL1A:     EQU   #00E2      ;VALOR REG. 1 SPRITE 16*16 SIN AUMENTO
   20 VAL1B:     EQU   #00E3      ;IGUAL PERO CON AUMENTO
   30 VAL5A:     EQU   #0036      ;VALOR REG. 5, DIR. ATRIBUTOS=#1B00
   40 VAL5B:     EQU   #007F      ;IGUAL PERO EN #3F80
   50 ;
   60 ;
   70 ;
   80 CTR:       DEFB  0          ;CONTADOR CONMUTACION
```

```
 90 ;
100 ;
110 ;
120 SPATSW:    LD    A,(CTR)        ;COGE CONTADOR CONMUT
ACION
130            INC   A              ;LO INCREMENTA
140            LD    (CTR),A        ;LO RESTITUYE
150            AND   1              ;COMPRUEBA BIT 0
160            JR    NZ,UNMAG       ;SI 1, SPRITE SIN AUM
ENTO
170            LD    B,VAL1B        ;VALOR REG. 1 CON AUM
ENTO
180            LD    C,1            ;NO. REGISTRO
190            CALL  #0047          ;ESCRIBE B A REG. C
200            LD    B,VAL5B        ;VALOR REG. 5,(#1B00)
210            LD    C,5
220            CALL  #0047
230            RET
240 ;
250 ;
260 ;
270 UNMAG:     LD    B,VAL1A
280            LD    C,1
290            CALL  #0047
300            LD    B,VAL5A
310            LD    C,5
320            CALL  #0047
330            JP    SPATSW
```

La técnica anterior nos permite, no sólo mantener sprites de dos tamaños diferentes simultáneamente en la pantalla, sino que, como tenemos dos tablas de atributos de sprite, podemos tener el doble del número normal de sprites en pantalla, hasta un máximo de 64. Como solamente una tabla de atributos se encuentra activa en el momento dado, podremos, por tanto, disponer de un máximo de 8 sprites en cada línea horizontal, suponiendo, claro está, que los 4 de una misma línea pertenezcan a la misma tabla de atributos.

Debe tenerse en cuenta que en aquellos sistemas de video con persistencia en el tubo particularmente breve, puede presentarse una vibración excesiva. Por desgracia, no se puede ganar siempre.

ACCESO RAPIDO AL VDP

Aunque el método de acceso a la VRAM proporcionado por el VDP, tiene la ventaja de no precisar el valioso espacio de direcciones de la CPU,

pero, a su vez, tiene el inconveniente de ser lento Nunca es completamente posible evadirse de esta realidad, aunque sus peores efectos pueden ser evitados. En lo que respecta al acceso VRAM, existe una regla fundamental que no debe ser traspasada:

Hay que hacer uso del registro de direcciones del VDP, y acceder a la VRAM secuencialmente.

Como el acceso secuencial a la VRAM requiere solamente una transferencia de datos, mientras que el acceso no secuencial requiere dos transferencias de datos para cargar el registro de dirección cada vez que se utiliza, el acceso secuencial es significativamente más rápido. Por ejemplo, cuando escribimos una rutina para desplazar verticalmente la pantalla (scroll), es ventajoso leer la pantalla completa -la tabla de nombres- y cargarla en un área intermedia de la RAM, y después volverla a colocar en la VRAM en masa.

Mediante este método un scroll puede realizarse en menos de 1/50 seg. mientras que una rutina que utilice acceso no secuencial a la VRAM, no lo conseguirá en menos de 3/50 seg.

Si se van a manipular gran cantidad de datos de la VRAM, se deben cargar en la RAM y ejecutar las manipulaciones aquí, antes de devolverlos a la VRAM secuencialmente.

Con referencia a la figura 6.2, puede verse que el acceso al VDP es mucho más rapido durante los 4300 microsegundos que siguen a la interrupción de retorno de cuadro. En aquellos casos en que la temporización sea extremadamente crítica, es interesante realizar el acceso a la VRAM mediante una rutina controlada por interrupciones.

En resumen, el acceso rápido a la RAM, requiere la optimización de los siguientes procesos:

En primer lugar, se debe acceder a la VRAM tan poco como sea posible. Para ello, se debe mantener una copia de los datos de la VRAM a modificar en la RAM direccionable por la CPU.

En segundo lugar, y siempre que sea posible, se debe acceder a la VRAM secuencialmente, el acceso no secuencial consume mucho tiempo y, si se piensa cuidadosamente, puede resultar innecesario.

Por último, si la situación es desesperada, se debe intentar acceder a la VRAM solamente durante el tiempo de retorno de cuadro, puesto que durante este tiempo, el acceso a la VRAM no se corta hasta que el VDP termine lo que estuviera finalmente haciendo.

CAPITULO 7

El Generador de Sonido Programable

LOS REGISTROS DE DATOS

El generador de sonido elegido para el sistema MSX es el AY-3-8910 (o equivalente) de General Instruments. Este circuito fue discutido, brevemente, en el capítulo 5. El dispositivo contiene 16 registros de lectura/escritura, que permiten al usuario producir tono y ruido en cualquiera de los tres canales de que dispone. Los registros de datos del circuito son los siguientes:

Registro 0: Ajuste fino de frecuencia de tono del canal A.
Registro 1: Ajuste grueso de frecuencia de tono del canal A.
Registro 2: Ajuste fino de frecuencia de tono del canal B.
Registro 3: Ajuste grueso de frecuencia de tono del canal B.
Registro 4: Ajuste fino de frecuencia de tono del canal C.
Registro 5: Ajuste grueso de frecuencia de tono del canal C.
Registro 6: Frecuencia de ruido.
Registro 7: Activación y direccionamiento de E/S.
Registro 8: Amplitud del canal A y activación de envolvente.
Registro 9: Amplitud del canal B y activación de envolvente.
Registro10: Amplitud del canal C y activación de envolvente.
Registro11: Ajuste fino de frecuencia de envolvente.
Registro12: Ajuste grueso de frecuencia de envolvente.
Registro13: Forma de envolvente.
Registro14: Almacenamiento de datos del puerto A.
Registro15: Almacenamiento de datos del puerto B.

LOS GENERADORES DE TONO (REGISTROS 0 á 5)

Cada canal dispone de dos registros de frecuencia de tono asociados a él. La frecuencia del sonido a generar se fija en función de su inverso, que es el período (en unidades de 8 microsegundos). El registro de ajuste fino contiene los 8 bits menos significativos de dicho periodo, mientras que el registro de ajuste grueso almacena los 4 bits más significativos. Para ordenar al circuito la producción del sonido de un canal determinado, debe ponerse a cero el bit correspondiente del registro de activación (registro 7).

EL GENERADOR DE RUIDO (REGISTRO 6)

Este registro controla el generador de ruido seudoaleatorio incluido en el circuito. Su salida puede mezclarse con cualquiera de los tres canales de sonido, según se requiera, anulando el bit correspondiente del registro 7. El período del generador de ruido se fija mediante los 5 bits menos significativos de este registro.

EL REGISTRO DE ACTIVACION (REGISTRO 7)

El registro 7 especifica si se debe producir tono, ruido o ambos, y si se deben incluir en la salida de cada uno de los canales. También determina si los dos puertos de E/S van a utilizarse en modo de entrada o salida. Los bits se asignan de la siguiente forma:

Bit 0: Si se pone a nivel 1 desactiva el tono del canal A
Bit 1: Si se pone a nivel 1 desactiva el tono del canal B.
Bit 2: Si se pone a nivel 1 desactiva el tono del canal C.
Bit 3: Si se pone a nivel 1 desactiva el ruido del canal A.
Bit 4: Si se pone a nivel 1 desactiva el ruido del canal B.
Bit 5: Si se pone a nivel 1 desactiva el ruido del canal C.
Bit 6: Si se pone a nivel 1 configura el puerto A como salida.
Bit 7: Si se pone a nivel 0 configura el puerto B como entrada.

En el sistema MSX, los puertos de E/S se utilizan para la lectura de los mandos de juegos, y por lo tanto, deben funcionar siempre en el modo de entrada.

CONTROL DE AMPLITUD (REGISTROS 8 á 10)

Cada canal dispone de su propio registro de control de amplitud. El bit 4 de este registro especifica si se va a utilizar envolvente generada por el circuito para el canal. Si ponemos el bit 4 a nivel 1, la amplitud del canal pasa a estar bajo el control del generador de envolvente. Si el bit se anula, la amplitud se controla mediante programa, con los bits 0-3, donde 0 no produce volumen y 15 es el volumen máximo.

GENERADOR DE ENVOLVENTE (REGISTROS 11 á 13)

El circuito de sonido tiene un único generador de envolvente, que puede utilizarse por uno o los tres canales de sonido, de acuerdo con el registro de control de amplitud de cada canal. Los bits 0 á 3 del registro 13 determinan la forma de la envolvente de una manera un tanto arcaica. La figura 7.1 muestra los valores necesarios para generar las ocho formas de envolvente disponibles. Otros valores diferentes, duplicarán la envolvente correspondiente al valor 9 ó 15.

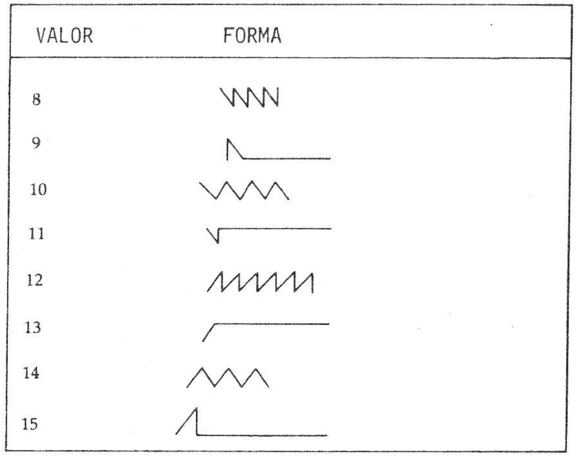
La longitud de cada subida o bajada, se determina mediante la frecuencia de envolvente.

El período de envolvente es un valor representado por 16 bits, contenido en el registro 11 (byte menos significativo) y el registro 12 (byte más significativo). El período se calcula en unidades de 128 microsegundos, y representa el tiempo entre cada escalón de la rampa. Dado que la rampa, sea subida o bajada, dispone de 16 escalones (equivalentes a los niveles de volumen comprendidos entre 0 y 15), el tiempo total de una rampa es de 1024 microsegundos por las unidades del período de envolvente. Por lo tanto, el período de envolvente fija la longitud de la rampa, aproximadamente en milisegundos.

LOS PUERTOS DE ENTRADA/SALIDA (REGISTROS 14 Y 15)

Los registros de Entrada/Salida se utilizan, por el sistema MSX, para la lectura de los mandos de juegos, por lo que no es conveniente emplearlos. Asímismo, se usan como memoria intermedia en el arranque o reposición del sistema.

Figura 7.1 Formas de Envolvente



PERIODOS DE NOTAS Y TONOS

El periodo de un tono determinado, puede calcularse a partir de la frecuencia, y mediante la fórmula:

Período=125000/Frecuencia

y la frecuencia de cualquier nota, en la escala correctamente templada, en el rango de las ocho octavas completas, se calcula a partir del Standard Internacional A, como:

Frecuencia=440*(2^(octava+(N-10)/12))

donde octava es el número de octava, siendo 0 la octava que contiene el "do" intermedio, -1 la octava inferior, 1 la octava superior, y asi, sucesivamente. N es el número de nota, 1 corresponde a "do", 2 a "do#", 3 a "re", etc.

Como el período es un valor entero, los valores calculados de las fórmulas anteriores, no producirán, exactamente, la frecuencia requerida. Sin embargo, los errores son muy pequeños, y suelen pasar desapercibidos.

La tabla siguiente recoge los valores correspondientes a la octava 0, calculados con las fórmulas anteriores, y un factor de error en cada

nota como porcentaje de la frecuencia real.

NOTA	FRECUENCIA	PERIODO	ERROR
do	261.626	478	0.046%
do#	277.183	451	0.007%
re	293.665	426	0.081%
re#	311.127	402	0.058%
mi	329.628	379	0.057%
fa	349.228	358	0.019%
fa#	369.994	338	0.046%
sol	391.995	319	0.037%
sol#	415.305	301	0.005%
la	440.000	284	0.032%
la#	466.164	268	0.055%
si	493.883	253	0.038%

Con las fórmulas anteriores, la siguiente rutina, en BASIC, calculará los períodos de las notas comprendidas en las ocho octavas completas, a partir de la cadena de entrada. Para simplificar la escritura de dicha cadena, el número de octava (el primer carácter de la cadena) debe variar entre 0 y 7 (octava baja a octava alta), en vez de -3 á 4.

La parte derecha de la cadena (la nota) puede tener hasta 2 caracteres de longitud, en cuyo caso el carácter de su derecha debe ser #.

```
1 '********************************
2 '***PERIODO DE NOTAS MUSICALES****
3 '********************************
4 'INTRODUZCA LA OCTAVA  Y LA NOTA          MUSICAL      EN LA ESC
ALA INGLESA
5 'OCTAVA 0-7  NOTAS C,C#,D,D#,E,F,         F#,G,G#,A,A#,B
10 INPUT A$
20 GOSUB 1000
30 PRINT A
40 GOTO 10
1000 O=VAL(LEFT$(A$,1))
1010 A$=MID$(A$,2)
1020 FOR NC=1 TO 12
1030 READ B$:IF B$=A$ THEN NO=NC
1040 NEXT NC
1050 RESTORE
1060 P=125000!/(440*(2^((O-3)+(NO-10)/12)))
1070 A=INT(P)
1080 RETURN
1090 DATA C,C#,D,D#,E,F,F#,G,G#,A,A#,B
```

Si ampliamos esta rutina, dispondremos, fácilmente, de un programa que nos proporcionará los datos de período de tono a utilizar en rutinas musicales de código máquina.

ACCESO AL GENERADOR PROGRAMABLE DE SONIDO EN EL SISTEMA MSX

Como los diseñadores del sistema MSX se reservan el derecho de alterar las especificaciones del soporte físico de las distintas series de ordenadores MSX, siempre que mantengan la compatibilidad de los programas, solamente se puede, de forma cierta, programar el generador de sonido mediante las llamadas del sistema operativo contenidos en la ROM del sistema. A continuación, veremos una descripción de estas rutinas:

DIRECCION	FUNCION
0090H	Esta rutina inicializa el generador de sonido, no requiere parámetros y puede modificar todos los registros.
0093H	Esta rutina escribe el dato contenido en E al registro de generador especificado por el acumulador, no retorna valores y deja los registros invariables.
0096H	Esta rutina lee un valor del registro especificado por el acumulador, almacenando la lectura en el mismo acumulador, y dejando invariables los restantes registros.

PROGRAMACION DEL GENERADOR DE SONIDO

La programación del generador de sonido precisa de una serie de operaciones de escritura a los registros para especificar el período de tono de un canal determinado, el período de ruido, cualquier forma de envolvente y su período, y/o la amplitud del canal correspondiente. Estas acciones deben realizarse con el canal al que se escribe desactivado (poniendo a nivel 1 el bit correspondiente del registro 7). Como ejemplo, la rutina siguiente genera la nota "do" intermedia en el canal A, con la forma de envolvente 8.

```
 10 MIDC:  LD   A,7
 20        CALL #0096     ;LECTURA REGISTRO ACTIVACION
 30        OR   9         ;DESCONEXION TONO Y RUIDO CANAL A
 40        LD   E,A
 50        LD   A,7
 60        CALL #0093     ;CARGA EN REGISTRO
 70        LD   E,1
 80        LD   A,1
 90        CALL #0093     ;PREPARACION AJUSTE GRUESO PERIODO CANAL A
100        LD   E,#00DE
110        LD   A,0
120        CALL #0093     ;AJUSTE FINO NOTA C=DO CENTRAL
130        LD   E,8
140        LD   A,13
150        CALL #0093     ;PREPARACION FORMA ENVOLVENTE
```



```
160        LD   E,15
170        LD   A,11
180        CALL #0093     ;AJUSTE FINO PERIODO ENVOLVENTE
190        LD   E,0
200        LD   A,12
210        CALL #0093     ;AJUSTE GRUESO PERIODO ENVOLVENTE
220        LD   E,16
230        LD   A,8
240        CALL #0093     ;ACTIVACION CONTROL POR ENVOLVENTE DEL CANAL A
250        LD   A,7
260        CALL #0096     ;LECTURA REGISTRO ACTIVACION
270        AND  #00FE     ;REPONE BANDERA DESACTIVACION TONO EN CANAL A
280        LD   E,A
290        LD   A,7
300        CALL #0093     ;CARGA NUEVO VALOR ACTIVACION
310        RET
```

Aunque la rutina anterior es un tanto trivial, desarrolla con efectividad, las técnicas de programación del generador de sonido. En el caso de un circuito, como el AY-3-8910, que dispone de unas posibilidades enormes, la única forma de conocerlo realmente es la experimentación. El resto de este capítulo desarrolla una serie de ejemplos con la intención de ayudar a conseguir mayor familiaridad con el circuito.

MUSICA EN TRES CANALES: EL ORDENADOR MUSICAL

La siguiente rutina, controlada mediante interrupciones, permite la ejecución en los tres canales musicales, independientemente de cualquier otro programa que se encuentre en funcionamiento. Los datos para los tres canales deben almacenarse en las direcciones asignadas a los símbolos C1DAT, C2DAT y C3DAT, como series de elementos de tres bytes. El primer byte de cada elemento es el valor de ajuste fino del período de nota, el segundo byte es el valor de ajuste grueso, y el tercer byte corresponde a la duración de la nota en unidades de 20 milisegundos. El final de la pieza musical se marca mediante el valor 255, situado en el byte de ajuste grueso del último dato del canal A. A lo largo de la ejecución del programa, y mientras suena la música, puede variarse el volumen de cada canal, escribiendo a las direcciones asignadas a los símbolos VOL1, VOL2 y VOL3. Para experimentar, con el generador de envolvente, éste puede activarse mediante la instrucción SOUND de BASIC, y después, escribiendo un valor de 16 a la dirección del volumen del canal relevante.

```
   10 INTHOK:    EQU   #F9DF             ;DIRECCION GANCHO INT
ERRUPCION
   20 PSGINI:    EQU   #0090             ;RUTINA DE S.O. INICI
ALIZACION PSG
   30 WRTPSG:    EQU   #0093             ;DIRECCION ESCRITURA
DATOS AL PSG
   40 RDPSG:     EQU   #0096             ;DIRECCION LECTURA DA
TOS DEL PSG
   50 C1DAT:     EQU   #B000             ;COMIENZO DATOS AJUST
E CANAL 1
   60 C2DAT:     EQU   #B400             ;CANAL 2
   70 C3DAT:     EQU   #B800             ;CANAL 3. PUEDEN ALTE
RARSE SEGUN SE PRECISE
   80            ORG   #E000
   90 START:     LD    HL,C1DAT
  100            LD    (C1PTR),HL
  110            LD    HL,C2DAT
  120            LD    (C2PTR),HL
  130            LD    HL,C3DAT
  140            LD    (C3PTR),HL        ;PREPARACION PUNTEROS
 DE DATOS
  150            LD    A,1
  160            LD    (C1CTR),A
  170            LD    (C2CTR),A
  180            LD    (C3CTR),A         ;PREPARACION CONTADOR
ES DURACION NOTA
  190            CALL  PSGINI            ;INICIALIZACION PSG
  200            PUSH  HL
  210            LD    HL,MUSROT         ;TOMA DIRECCION RUTIN
A MUSICA
  220            LD    A,L
  230            LD    (INTHOK+1),A      ;CARGA DIRECCION BAJA
  240            LD    A,H
  250            LD    (INTHOK+2),A      ;CARGA DIRECCION ALTA
  260            POP   HL
  270            LD    A,#00C3           ;TOMA INSTRUCCION Z80
 "JP"
  280            LD    (INTHOK),A        ;LA CARGA EN GANCHO I
NTERRUPCION
  290            RET                     ;QUEDA PREPARADA LA I
NTERRUPCION PARA EJECUTAR RUTINA MUSICA
  300 C1PTR:     DEFW  0
  310 C2PTR:     DEFW  0
  320 C3PTR:     DEFW  0
  330 C1CTR:     DEFB  0
  340 C2CTR:     DEFB  0
  350 C3CTR:     DEFB  0
  360 VOL1:      DEFB  15
  370 VOL2:      DEFB  15
  380 VOL3:      DEFB  15                ;PREPARACION TABLA VA
RIABLES
  390 MUSROT:    PUSH  AF                ;GUARDA ESTADO VDP
  400            LD    A,(C1CTR)         ;COMIENZO RUTINA MUSI
CA
  410            DEC   A                 ;DISMINUYE DURACION N
OTA
  420            LD    (C1CTR),A
  430            OR    A                 ;HA TERMINADO LA NOTA
?
  440            JR    NZ,CHANB          ;SI NO TOCA CANAL 2
  450            LD    A,7               ;SI SE TERMINO
```

```
  460            CALL  RDPSG
  470            OR    1
  480            LD    E,A
  490            LD    A,7
  500            CALL  WRTPSG            ;APAGA EL CANAL 1
  510            LD    IX,(C1PTR)        ;CARGA PUNTERO DATOS
EN IX
  520            LD    E,(IX+0)          ;TOMA VALOR DE AJUSTE
 FINO
  530            LD    A,0
  540            CALL  WRTPSG            ;LO ESCRIBE EN REGIST
RO 0
  550            LD    A,(IX+1)          ;TOMA VALOR AJUSTE GR
UESO
  560            CP    255               ;ES EL FIN DEL MARCAD
OR DE AJUSTE?
  570            JR    Z,START           ;SI, REINICIALIZA
  580            LD    E,A
  590            LD    A,1
  600            CALL  WRTPSG            ;SI NO LO ESCRIBE EN
REGISTRO 1
  610            LD    A,(IX+2)          ;PREPARA CONTADOR DUR
ACION DE NOTA
  620            LD    (C1CTR),A
  630            LD    A,(VOL1)          ;TOMA VOLUMEN CANALA
  640            LD    E,A
  650            LD    A,8
  660            CALL  WRTPSG            ;LO ESCRIBE EN REGIST
RO VOLUMEN CANAL 1
  670            LD    A,7
  680            CALL  RDPSG
  690            AND   #00FE
  700            LD    E,A
  710            LD    A,7
  720            CALL  WRTPSG            ;REACTIVA TONO EN CAN
AL 1
  730            INC   IX
  740            INC   IX
  750            INC   IX                ;APUNTA A LA SIGUIENT
E NOTA
  760            LD    (C1PTR),IX        ;GUARDA PUNTERO
  770 CHANB:     LD    A,(C2CTR)         ;LO MISMO PARA EL CAN
AL 2
  780            DEC   A
  790            LD    (C2CTR),A
  800            OR    A
  810            JR    NZ,CHANC
  820            LD    A,7
  830            CALL  RDPSG
  840            OR    2
  850            LD    E,A
  860            LD    A,7
  870            CALL  WRTPSG
  880            LD    IX,(C2PTR)
  890            LD    E,(IX+0)
  900            LD    A,2
  910            CALL  WRTPSG
  920            LD    E,(IX+1)
  930            LD    A,3
  940            CALL  WRTPSG
  950            LD    A,(VOL2)
```

```
 960            LD    E,A
 970            LD    A,9
 980            CALL  WRTPSG
 990            LD    A,7
1000            CALL  RDPSG
1010            AND   #00FD
1020            LD    E,A
1030            LD    A,7
1040            CALL  WRTPSG
1050            LD    A,(IX+2)
1060            LD    (C2CTR),A
1070            INC   IX
1080            INC   IX
1090            INC   IX
1100            LD    (C2PTR),IX
1110 CHANC:     LD    A,(C3CTR)
1120            DEC   A
1130            LD    (C3CTR),A
1140            OR    A
1150            JR    NZ,ENDMUS
1160            LD    A,7
1170            CALL  RDPSG
1180            OR    4
1190            LD    E,A
1200            LD    A,7
1210            CALL  WRTPSG
1220            LD    IX,(C3PTR)
1230            LD    E,(IX+0)
1240            LD    A,4
1250            CALL  WRTPSG
1260            LD    E,(IX+1)
1270            LD    A,5
1280            CALL  WRTPSG
1290            LD    A,(IX+2)
1300            LD    (C3CTR),A
1310            LD    A,(VOL3)
1320            LD    E,A
1330            LD    A,10
1340            CALL  WRTPSG
1350            LD    A,7
1360            CALL  RDPSG
1370            AND   #00FB
1380            LD    E,A
1390            LD    A,7
1400            CALL  WRTPSG
1410            INC   IX
1420            INC   IX
1430            INC   IX
1440            LD    (C3PTR),IX
1450 ENDMUS:    POP   AF                    ;RESTAURA ESTADO VDP
1460            RET
```


EFECTOS DE SONIDO EN EL AY-3-8910

En la producción de efectos de sonido no hay, en absoluto, ningún sustituto de la experimentación. Si bien es cierto que el PSG puede producir, virtualmente, cualquier sonido imaginable, en la mayoría de los casos, hay que emplear algunas horas probando varios valores en los registros, antes de conseguir siquiera una ligera aproximación al sonido deseado. Este procedimiento se realiza mejor en BASIC, utilizando la instrucción SOUND, con la que rápida y fácilmente se puede cambiar cualquier registro del PSG. Habiendo encontrado los valores de registros necesarios, se pueden codificar fácilmente las rutinas precisas para cargar los valores en el generador.

Por ejemplo, el siguiente programa en BASIC produce el sonido de un disparo:

```
10 SOUND 6,15:SOUND 7,7
20 SOUND 8,16:SOUND 9,16:SOUND 10,16
30 SOUND 11,0:SOUND 12,16:SOUND 13,0
```

Esta misma rutina puede escribirse en código Z-80 de la siguiente forma:

```
 10 GUN:        LD    B,6
 20             LD    HL,GUNTBL
 30 GUNLP:      LD    E,(HL)
 40             LD    A,B
 50             CALL  #0093
 60             INC   HL
 70             INC   B
 80             LD    A,B
 90             CP    14
100             JR    NZ,GUNLP
110             RET
120 GUNTBL:     DEFB  15,7,16,16,16,0,16,0
```

UTILIZACION DEL PUERTO DE SONIDO DE UN BIT

Además de la generación de sonido mediante el circuito del generador programable, el sistema MSX reserva un bit del puerto C del interfaz programable para periféricos 8255, para la generación de sonido, con programa. Una llamada del sistema operativo ubicada en 0135H, toma el valor contenido en el acumulador, si este valor es 0, desconecta el

puerto de salida de un bit, en caso contrario, lo conecta. Alternando la conexión y desconexión rápidamente, mediante programa, se produce sonido, como demuestra la siguiente rutina:

```
SFTSND:   LD    A,0
          CALL  #0135
          LD    A,1
          CALL  #0135
          JR    SFTSND
```

Al ejecutarla, deberá oirse un tono agudo. Se puede experimentar la conexión y desconexión del puerto, a diferentes velocidades, para ver lo que sucede.

CAPITULO 8

Operaciones de Entrada/Salida.

El sistema MSX dispone de un extenso conjunto de rutinas que realizan funciones tales como lectura y exploración del teclado, lectura de mandos de juegos, raquetas e impresión en pantalla. Esta capítulo estudia, en primer lugar, el manejo de los mandos de juegos, y a continuación, las operaciones de entrada/salida de teclado, pantalla y selección de ranuras.

ENTRADA/SALIDA DE MANDOS DE JUEGOS, RAQUETAS Y TABLEROS DIGITALIZADORES

El sistema MSX admite hasta dos mandos de juegos, y considera las teclas de cursor y la barra espaciadora como un tercer mando. Las siguientes rutinas se utilizan para la lectura de los mandos de juegos y disparadores.

OD5H — Lee el estado actual del mando de juego especificado por el acumulador (0 á 2, donde 0 se refiere a las teclas de cursor). Carga el acumulador con un valor que varía entre 0 y 8, según el diagrama siguiente. El valor 0 indica que el mando de juego está centrado. Esta rutina puede modificar todos los registros.

```
              1
          8   |   2
      7   -   0   -   3
          6   |   4
              5
```

OD8H — Esta rutina lee el disparador del mando de juegos especificado por el identificador de mando de juegos cargado en el acumulador. Entrega el valor 0 en el mismo acumulador, si el disparador no se encuentra pulsado, y 255 si lo está. Solamente se modifica el par de registros AF. Si el identificador es 0, se comprueba la barra de espacio.

Los puertos de mandos de juegos están diseñados, también, para permitir la lectura de raquetas de juego. Se pueden conectar hasta seis raquetas en cada puerto, haciendo un total de doce, que se numeran de 1 á 12. Las raquetas con número impar se conectan al puerto 1, y las pares, al 2.

0DEH Esta rutina lee la raqueta de juegos especificada por el acumulador, y entrega un valor comprendido entre 0 y 255 (dependiendo de la posición de la paleta) en el acumulador. Puede modificar todos los registros.

Por último, los puertos de mandos de juegos pueden emplearse para leer tableros digitalizadores compatibles con el NEC PC-6051. La siguiente rutina se emplea para leer el estado de dicho tablero:

0BDH Esta rutina recibe un identificador en el acumulador y entrega un valor del tablero, modificando todos los registros. Los identificadores y valores recibidos son los siguientes:

ID	RETORNO
0	255 si el tablero del puerto 1 está pulsado, 0 en caso contrario.
1	Coordenada X del punto presionado en el tablero 1.
2	Coordenada Y del anterior.
3	255 si el interruptor del tablero 1 está conectado, 0 en caso contrario.
4-7	Los identificadores 4-7 se corresponden con los anteriores para el tablero en el puerto 2.

ENTRADA/SALIDA DE TECLADO Y PANTALLA

Se dispone de las siguientes rutinas para acceder al teclado y pantalla:

09CH	Esta rutina comprueba el estado de la memoria intermedia de teclado. Pone a nivel 1 la bandera Z, si la memoria intermedia está vacía, y modifica el par de registros AF.
09FH	Esta rutina toma un carácter de la memoria intermedia de teclado, si ésta se encuentra vacía, espera a que se teclee un carácter. Carga el valor ASCII del carácter en el acumulador, y modifica el par AF.
0A2H	Imprime el carácter cuyo código se encuentra en el acumulador, en la posicion actual del cursor, dejando todos los registros sin alteración.
0C0H	Produce un sonido (bip), equivalente a CHR$(7). Puede modificar todos los registros.
0C3H	Borra la pantalla, modificando los pares AF, BC y DE.
0C6H	Sitúa el cursor de pantalla en la columna contenida en H, y en la fila L, modificando el par AF.
0CCH	Borra el contenido de las teclas de función, modificando todos los registros.
0CFH	Muestra el contenido de las teclas de función, modificando todos los registros.

El uso de estas rutinas es bastante sencillo, y la programación de aplicaciones de usuario no debe suponer grandes problemas, puesto que se dispone de las funciones básicas precisas de E/S a consola; por

ejemplo, una simple rutina de entrada y visualización, sería:

```
   10            LD    HL,0
   20            CALL  #00C6              ;CURSOR EN ESQUINA IN
FERIOR DERECHA
   30            CALL  #00C3              ;BORRA PANTALLA
   40  INLP:     CALL  #009F              ;COGE CARACTER
   50            CALL  #00A2              ;LO IMPRIME
   60            CP    13
   70            JR    NZ,INLP            ;LO REPITE HASTA QUE
SE PULSA RETURN
   80            RET                      ;SI NO VUELVE A BASIC
```

En el ejemplo siguiente, la rutina pide una palabra clave por teclado, antes de permitir la continuación de la ejecución:

```
   10 PSSWD:     LD    HL,PSWORD          ;CLAVE EN HL
   20 PWLP:      CALL  #009F              ;COGE UN CARACTER
   30            CP    (HL)               ;ES LA CLAVE?
   40            JR    Z,PWSKP            ;SI, SIGUE EL BUCLE
   50            LD    A,(HL)
   60            CP    "&"                ;FIN DE CLAVE?
   70            JR    Z,GO               ;SI, SIGUE PROGRAMA
   80            JP    #0000              ;NO, SE REPONE EL SIS
TEMA
   90 PWSKP:     INC   HL                 ;INCREMENTA PUNTERO C
LAVE
  100            CALL  #00A2              ;IMPRIME CARACTER PRE
VIO
  110            JR    PWLP               ;SIGUIENTE CARACTER
  120 PSWORD:    DEFM  "PALABRA CLAVE&"
  130 GO:        RET                      ;INSERTAR RESTANTES R
UTINAS A PARTIR DE AQUI
```

SELECCION DE RANURAS

Como el PPI 8255 puede no estar direccionado en la misma posición, en distintos ordenadores MSX, se han previsto dos rutinas para facilitar la lectura y escritura del puerto A del mismo (es decir, el registro de selección de ranura primaria):

0138H	Esta rutina lee el valor actual del registro de selección de ranura, cargando dicho valor en el acumulador, y dejando inalterados los restantes registros.
013BH	Escribe el valor contenido en el acumulador al registro de selección de ranura primaria, dejando inalterados los restantes registros.

Si se desea seleccionar las páginas 2 y 3 de la ranura 2, dejando invariables las páginas 0 y 1, se podría utilizar la siguiente rutina:

```
   10            CALL  #0138              ;LEE REGISTRO SELECCI
ON RANURA
   20            AND   15                 ;ENMASCARA BITS PAGS.
 0,1
   30            OR    %10100000          ;ENMASCARA PARA SELEC
CION RANURA 2, PAGS. 2,3
   40            CALL  #013B              ;ESCRIBE NUEVO VALOR
```

Existen otras dos rutinas del sistema operativo, relacionadas con el PPI 8255, la primera está situada en 0132H y acepta un valor en el

acumulador (0 ó distinto de 0), que si es 0, apaga la lámpara de mayúsculas (CAPS), y en caso contrario, la enciende. La segunda rutina se utiliza para explorar el teclado.

EXPLORACION DEL TECLADO: COMPROBACION DE TECLAS INDIVIDUALES

0141H La rutina de esta posición explora la fila de la matriz de teclado (veáse figura 8.1), especificada por el valor contenido por el acumulador (0 á 9), y retorna con un valor en el acumulador tal que el bit correspondiente a una de las columnas de la matriz de teclado está a cero si la tecla perteneciente a dicha columna de la fila explorada, está pulsada. Si no está pulsada ninguna tecla de la fila, la rutina retornará el valor 255.

Figura 8.1 La matriz de teclado

COLUMNAS

FILAS	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	+	[		¥	-	=	9	8
2	B	A	-	/	>	<	]	*
3	J	I	H	G	F	E	D	C
4	R	Q	P	O	N	M	L	K
5	Z	Y	X	W	V	U	T	S
6	F3	F2	F1	CODE	CAP	GRAPH	CTRL	SHIFT
7	RETURN	SELECT	BS	STOP	TAB	ESC	FS	F4
8	→	↓	↑	←	DEL	INS	HOME	SPACE
9								

Como explicación de lo anterior, considérese la siguiente rutina, que explora el teclado hasta que las teclas Z y X sean pulsadas conjuntamente.

```
 10 ZXCHK:     LD   A,5               ;EXPLORA FILA 5 MATRI
Z TECLADO
 20           CALL #0141
 30           AND  %10100000         ;COMPRUEBA SI LOS BIT
S 7 Y 5 ESTAN AMBOS A 0
 40           JR   NZ,ZXCHK          ;SI NO, REPITE
 50           RET
```

APENDICES

A ... Códigos de carácter

B ... Códigos de color

C ... Tabla de VRAM

D ... Instrucciones Z-80

E ... TI 9929 A VDP

F ... GI AY-3-8910 PSG

APENDICE A

CODIGOS DE CARACTER

Reproducido con permiso de Toshiba UK Limited.

4 bits más significativos →

← 4 bits menos significativos

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			Blank (Space)	Ø	@	P	`	p	Ç	É	á	Ã			α	≡
1			!	1	A	Q	a	q	ü	æ	í	ã			β	±
2		INS	"	2	B	R	b	r	é	Æ	ó	Ĩ			Γ	≥
3			#	3	C	S	c	s	â	ô	ú	ĩ			Π	≤
4			$	4	D	T	d	t	ä	ö	ñ	Õ			Σ	⌠
5			%	5	E	U	e	u	à	ò	Ñ	õ			σ	⌡
6			&	6	F	V	f	v	å	û	ª	Ũ			μ	÷
7	BL		'	7	G	W	g	w	ç	ù	º	ũ			Υ	≈
8	BS	SELECT	(	8	H	X	h	x	ê	ÿ	¿	Ĳ		△	Φ	°
9	TAB		)	9	I	Y	i	y	ë	Ö	⌐	ĳ		╪	Θ	∙
A	LF		*	:	J	Z	j	z	è	Ü	¬	¾		ω	Ω	·
B	HOME	ESC	+	;	K	[	k	{	ï	¢	½	∽			δ	√
C	CLS	→	,	<	L	\	l	¦	î	£	¼	◇			∞	^{n}
D	CR	←	-	=	M	]	m	}	ì	¥	¡	‰			φ	^{2}
E		↑	.	>	N	^	n	~	Ä	₧	«	¶			∈	
F		↓	/	?	O	_	o	Blank (DEL)	Å	ƒ	»	§			∩	Blank (FF)

← números hexadecimales

↑ números hexadecimales

APENDICE B

CODIGOS DE COLOR

0 ... Transparente
1 ... Negro
2 ... Verde
3 ... Verde claro
4 ... Azul oscuro
5 ... Azul claro
6 ... Rojo oscuro
7 ... Azul celeste
8 ... Rojo
9 ... Rojo claro
10 ... Amarillo oscuro
11 ... Amarillo claro
12 ... Verde oscuro
13 ... Magenta
14 ... Gris
15 ... Blanco

APENDICE C

Tabla de VRAM

	MODO TEXTO 40 COLUMNAS	MODO TEXTO 32 COLUMNAS	MODO GRAFICO ALTA RES.	MODO GRAFICO MULTICOLOR
TABLA NOMBRES	0	6144	6144	2048
TABLA PATRONES	2048	0	0	0
TABLA COLOR	-	8192	8192	-
TABLA ATRIBUTOS SPRITE	-	6912	6912	6912
TABLA PATRONES SPRITE	-	14336	14336	14336

APENDICE D

Instrucciones Z-80

Los signos situados debajo de cada bandera, tienen los siguientes significados:

R: La bandera se altera como consecuencia de la operación.
0: La bandera se repone (nivel 0).
1: La bandera se pone (nivel 1).
C: El valor de la bandera C (exceso) se copia en la bandera H.

Las restantes abreviaturas se encuentran en la lista al final de este Apéndice.

	BANDERAS					
	S	Z	H	N	P	C
ADC HL,ss	R	R	R	0	R	R
ADC A,s	R	R	R	0	R	R
ADD A,n	R	R	R	0	R	R
ADD A,r	R	R	R	0	R	R
ADD A,(HL)	R	R	R	0	R	R
ADD A,(IX+d)	R	R	R	0	R	R
ADD A,(IY+d)	R	R	R	0	R	R
ADD HL,ss	-	-	R	0	-	R
ADD IX,pp	-	-	R	0	-	R
ADD IY,rr	-	-	R	0	-	R
AND s	R	R	1	0	R	0
BIT b,(HL)	-	R	1	0	-	-
BIT b,(IX+d)	-	R	1	0	-	-
BIT b,(IY+d)	-	R	1	0	-	-
BIT b,r	-	R	1	0	-	-
CALL cc,nn	-	-	-	-	-	-
CALL nn	-	-	-	-	-	-
CCF	-	-	C	0	-	R
CP s	R	R	R	1	R	R
CPD	R	R	R	1	R	-
CPDR	R	R	R	1	R	-

	BANDERAS					
	S	Z	H	N	P	C
CPI	R	R	R	1	R	-
CPIR	R	R	R	1	R	-
CPL	-	-	1	1	-	-
DAA	R	R	R	-	R	R
DEC m	R	R	R	1	R	-
DEC IX	-	-	-	-	-	-
DEC IY	-	-	-	-	-	-
DEC ss	-	-	-	-	-	-
DI	-	-	-	-	-	-
DJNZ e	-	-	-	-	-	-
EI	-	-	-	-	-	-
EX (SP),HL	-	-	-	-	-	-
EX (SP),IX	-	-	-	-	-	-
EX (SP),IY	-	-	-	-	-	-
EX AF,AF'	R	R	R	R	R	R
EX DE,HL	-	-	-	-	-	-
EXX	-	-	-	-	-	-
HALT	-	-	-	-	-	-
IM 0	-	-	-	-	-	-
IM 1	-	-	-	-	-	-
IM 2	-	-	-	-	-	-

IN A,(N)	-	-	-	-	-	-
IN R,(C)	R	R	R	0	R	-
INC (HL)	R	R	R	0	R	-
INC IX	-	-	-	-	-	-
INC (IX+d)	R	R	R	0	R	-
INC IY	-	-	-	-	-	-
INC (IY+d)	R	R	R	0	R	-
INC r	R	R	R	0	R	-
INC ss	-	-	-	-	-	-
IND	-	R	-	1	-	-
INDR	-	1	-	1	-	-
INI	-	R	-	1	-	-
INIR	-	1	-	1	-	-
JP (HL)	-	-	-	-	-	-
JP (IX)	-	-	-	-	-	-
JP (IY)	-	-	-	-	-	-
JP cc,nn	-	-	-	-	-	-
JP nn	-	-	-	-	-	-
JR C,e	-	-	-	-	-	-
JR e	-	-	-	-	-	-
JR NC,e	-	-	-	-	-	-
JR NZ,e	-	-	-	-	-	-
JR Z,e	-	-	-	-	-	-
LD A,(BC)	-	-	-	-	-	-
LD A,(DE)	-	-	-	-	-	-
LD A,I	R	R	0	0	R	-
LD A,(nn)	-	-	-	-	-	-
LD A,R	R	R	0	0	R	-
LD (BC),A	-	-	-	-	-	-
LD (DE),A	-	-	-	-	-	-
LD (HL),n	-	-	-	-	-	-
LD dd,nn	-	-	-	-	-	-
LD HL,(nn)	-	-	-	-	-	-
LD (HL),r	-	-	-	-	-	-
LD I,A	-	-	-	-	-	-
LD IX,nn	-	-	-	-	-	-
LD IX,(nn)	-	-	-	-	-	-
LD (IX+d),n	-	-	-	-	-	-
LD (IX+d),r	-	-	-	-	-	-
LD IY,nn	-	-	-	-	-	-
LD IY,(nn)	-	-	-	-	-	-
LD (IY+d),n	-	-	-	-	-	-
LD (IY+d),r	-	-	-	-	-	-
LD (nn),A	-	-	-	-	-	-
LD (nn),dd	-	-	-	-	-	-
LD (nn),HL	-	-	-	-	-	-
LD (nn),IX	-	-	-	-	-	-
LD (nn),IY	-	-	-	-	-	-
LD R,A	-	-	-	-	-	-
LD r,(HL)	-	-	-	-	-	-
LD r,(IX+d)	-	-	-	-	-	-
LD r,(IY+d)	-	-	-	-	-	-
LD r,n	-	-	-	-	-	-
LD r,r'	-	-	-	-	-	-
LD SP,HL	-	-	-	-	-	-
LD SP,IX	-	-	-	-	-	-
LD SP,IY	-	-	-	-	-	-

LDD	-	-	0	0	R	-
LDDR	-	-	0	0	R	-
LDI	-	-	0	0	R	-
LDIR	-	-	0	0	R	-
NEG	R	R	R	1	R	R
NOP	-	-	-	-	-	-
OR s	R	R	0	0	R	0
OTDR	-	1	-	1	-	-
OTIR	-	1	-	1	-	-
OUT (C),r	-	-	-	-	-	-
OUT (n),A	-	-	-	-	-	-
OUTD	-	R	-	1	-	-
OUTI	-	R	-	1	-	-
POP IX	-	-	-	-	-	-
POP IY	-	-	-	-	-	-
POP qq	-	-	-	-	-	-
PUSH IX	-	-	-	-	-	-
PUSH IY	-	-	-	-	-	-
PUSH qq	-	-	-	-	-	-
RES b,m	-	-	-	-	-	-
RET	-	-	-	-	-	-
RET cc	-	-	-	-	-	-
RETI	-	-	-	-	-	-
RETN	-	-	-	-	-	-
RL m	R	R	0	0	R	R
RLA	-	-	0	0	-	R
RLC (HL)	R	R	0	0	R	R
RLC (IX+d)	R	R	0	0	R	R
RLC (IY+d)	R	R	0	0	R	R
RLC,r	R	R	0	0	R	R
RLCA	-	-	0	0	-	R
RLD	R	R	0	0	R	R
RR m	R	R	0	0	R	R
RRA	-	-	0	0	-	R
RRC m	R	R	0	0	R	R
RRCA	-	-	0	0	-	R
RRD	R	R	0	0	R	R
RST p	-	-	-	-	-	-
SBC A,s	R	R	R	1	R	R
SBC HL,ss	R	R	R	1	R	R
SCF	-	-	0	0	-	1
SET b,(HL)	-	-	-	-	-	-
SET b,(IY+d)	-	-	-	-	-	-
SET b,r	-	-	-	-	-	-
SLA m	R	R	0	0	R	R
SRA m	R	R	0	0	R	R
SRL m	R	R	0	0	R	R
SUB s	R	R	R	1	R	R
XOR s	R	R	R	1	R	R
SET b,(IX+d)	-	-	-	-	-	-

ABREVIATURAS:

r	Registro: A, B, C, D, E, H o L
n	Un valor comprendido entre 0 y 255, entero y sin signo
s	r, n, (HL), (IX+d) o (IY+d)
m	A, B, C, E, H, L, (HL), (IX+d), (IY+d)
dd	BC, DE, HL, SP |LD HL,(dd) LD (nn),dd|
nn	Un valor comprendido entre 0 y 65535, entero
qq	BC, DE, HL, AF |POP qq PUSH qq|
ss	BC, DE, HL, SP |ADC HL,ss ADD HL,ss DEC ss INC ss SBC HL,ss|
pp	BC, DE, IX, SP |ADD IX,pp|
rr	BC, DE, IY, SP |ADD IY,rr|
b	Bit 0 á 7
e	Un byte de desplazamiento, con signo comprendido entre -126 a 129
p	Una dirección de la página 0, incluida en la instrucción del procesador RESTART. Debe ser múltiplo de 8, comprendida entre 0 y 56, inclusive.
cc	Código de condición: C, NC, Z, NZ, P, M, PE, PO

APENDICE E

TI 9929 A VDP

(Reproducido con permiso de Texas Instruments Ltd.)

ARQUITECTURA DEL VDP

El procesador de pantalla de video TMS 9118 está diseñado para permitir un enlace sencillo entre un microprocresador y una televisión o un monitor de color con exploración por barrido. Los procesadores de video TMS 9128 y 9129 están diseñados, a su vez, como enlaces sencillos entre un microprocesador y un monitor RGB o un codificador de video, que produzca la señal de video para controlar el monitor de video. La figura E.1 es un diagrama de bloques de las partes de la Arquitectura del VDP que conectan con la CPU, VRAM y televisión en color.

INTERFAZ CON LA CPU

El VDP se conecta con la CPU utilizando un bus de datos bidireccional de 8 bits, tres líneas de control y una interrupción. A través de estos interfaces, la CPU puede realizar cuatro operaciones:

1. Escritura a uno de los 8 registros de escritura del VDP
2. Lectura del registro de estado del VDP
3. Escritura de bytes de datos de pantalla a la VRAM
4. Lectura de bytes de datos de pantalla desde la VRAM

Cada una de estas operaciones requiere la transferencia de uno o más datos, a través del interfaz del bus de datos entre la CPU y el VDP. La interpretación de la transferencia de datos se determina mediante las tres líneas de control del VDP.

La CPU se puede comunicar con el VDP en forma simultánea y asíncrona, con referencia a las operaciones de refresco de pantalla realizadas por el VDP. El VDP realiza la gestión de memoria, y permite el acceso de la CPU a la VRAM en intervalos periódicos, incluso a la mitad de un barrido de líneas.

BUS DE DATOS CPU/VDP

La CPU transfiere datos entre ella misma y el VDP, a través de un bus de datos bidireccional. Los 8 bits de este bus se representan por

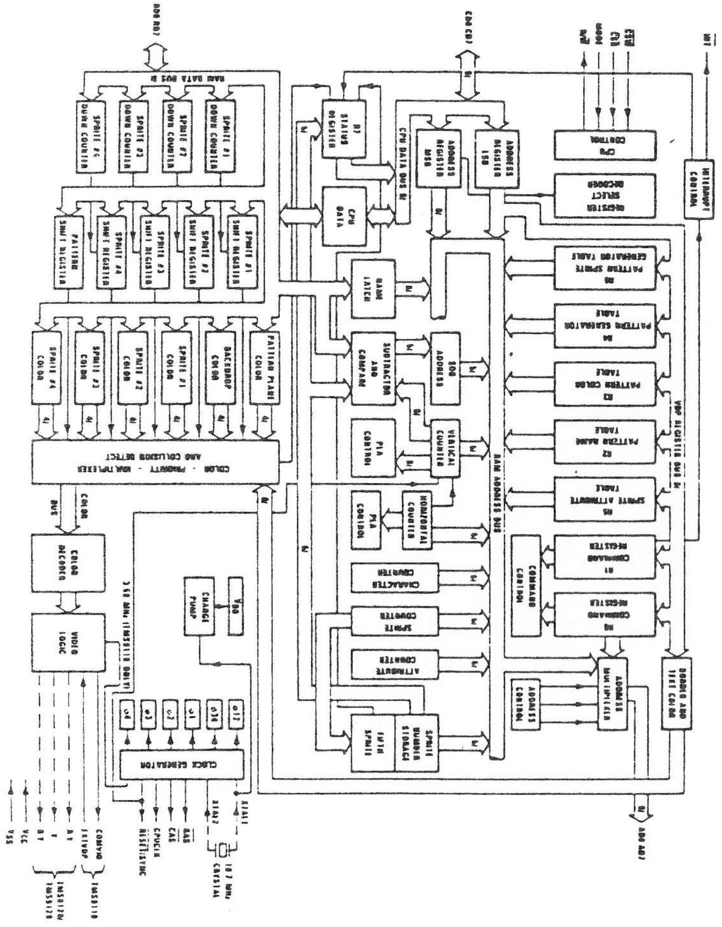


Figura E.1 Diagrama de bloques del VDP

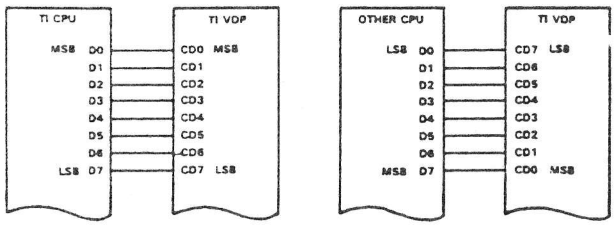


Figura E.2 Conexión CPU/VDP

CD0 (bit más significativo) a CD7 (bit menos significativo), como puede verse en la figura E.1 y E.2.

A lo largo de este manual, el interfaz CPU/VDP se describe asumiendo la utilización de circuitos TI. Otros fabricantes de CPUs asignan la línea D0 como bit menos significativo y D7 como bit más significativo. La conexión correcta se muestra en la figura E.2. Si la conexión no se realiza correctamente, no habrá visualización de pantalla.

SEÑALES DE CONTROL DEL INTERFAZ DE LA CPU

Las entradas CSW, CSR y MODE controlan el tipo y dirección de transferencias de datos. CSW selecciona el modo de escritura de CPU a VDP. Cuando está activa (nivel bajo), los 8 bits en CD0-CD7 se envían al VDP. CSR selecciona el modo de lectura por la CPU del VDP. Cuando está activo (nivel bajo), el VDP envía los 8 bits en CD0-CD7 a la VDP. CSW y CSR nunca deben estar, simultáneamente, a nivel bajo. Si ambos están bajos, el VDP envía datos no válidos a través de CD0-CD7.

La entrada MODE determina la fuente o destino de la transferencia, lectura o escritura, de los datos. Esta entrada está normalmente conectada a una línea de direcciones de bajo orden de la CPU.

REGISTROS DEL VDP.

El VDP tiene 8 registros, que solamente admiten escritura, y un registro de estado que sólo permite lectura. Los registros de escritura controlan las acciones del VDP y determinan la asignación de la VRAM. Los registros de estado contienen las banderas de interrupción, coincidencia de sprites y estado del quinto sprite. Más adelante se detallan las caracteristicas de todos los registros.

Cada uno de los 8 registros de escritura pueden cargarse utilizando dos transferencias de datos de 8 bits desde la CPU. La tabla E.3 describe el formato preciso de los dos bytes. El primer byte transferido es el byte de datos, y el segundo byte, controla el destino. El bit más significativo del segundo byte debe ser 1.

Los siguientes cuatro bits son 0, y los tres bits inferiores constituyen el número de registro de destino (0-7). La entrada MODE estará a nivel alto para ambas transferencias de bytes.

Si los dos bits más significativos del segundo byte indican que se está procesando una escritura a registro, el primer byte no se interpreta como una dirección de la CPU.

ESCRITURA DE LA CPU A LA VRAM

La CPU transfiere datos a la VRAM, a través del VDP, utilizando un registro de direcciones de incremento automático y con 14 bits de longitud. La preparación del registro de direcciones requiere transferencias de 2 bytes. Cada preparación de dirección consume dos microsegundos, con un total de 4 microsegundos (veáse tabla E.3). Se requiere una transferencia de un byte para escribir el dato al byte direccionado de la VRAM. El tiempo necesario depende del MODE. El registro de dirección se incrementa automáticamente. La escritura secuencial

a la VRAM requiere, únicamente, transferencias de un byte, puesto que el registro de dirección ya está preparado. Durante la carga del registro de dirección, los dos bits más significativos del segundo byte de dirección, deben ser 0 y 1, respectivamente. MODE estará a nivel alto para ambas transferencias de direcciones, y a nivel bajo para la transferencia del dato. CSW se utiliza en todas las transferencias para sincronizar los 8 bits en el VDP. (Véase tabla E.3).

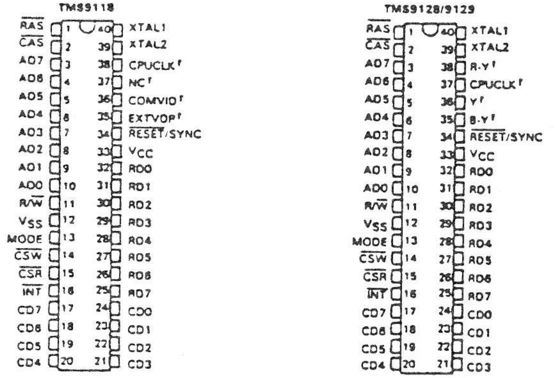


Figura E.3 Asignaciones de patillas de los VDP TMS 9118/9128/9129

OPERACION	0	1	2	3	4	5	6	7	CSW	CSR	MODE	TIEMPO
Escritura a registro del VDP												
Byte 1 esc. dato	D0	D1	D2	D3	D4	D5	D6	D7	0	1	1	2 us
Byte 2 selec. reg.	1	0	0	0	0	RS0	RS1	RS2	0	1	1	2 us
Escritura a VRAM												
Byte 1 dirección	A6	A7	A8	A9	A10	A11	A12	A13	0	1	1	2 us
Byte 2 dirección	0	1	A0	A1	A2	A3	A4	A5	0	1	1	2 us
Byte 3 esc. dato	D0	D1	D2	D3	D4	D5	D6	D7	0	1	0	-
Lectura de registro de estado												
Byte 1 lec. dato	D0	D1	D2	D3	D4	D5	D6	D7	1	0	1	2 us
Lectura de VRAM												
Byte 1 dirección	A6	A7	A8	A9	A10	A11	A12	A13	0	1	1	2 us
Byte 2 dirección	0	0	A0	A1	A2	A3	A4	A5	0	1	1	-
Byte 3 lec. dato	D0	D1	D2	D3	D4	D5	D6	D7	1	0	0	-

Tabla E.3 Transferencias de datos CPU/VDP y Secuencias de preparación.

LECTURA POR LA CPU DEL REGISTRO DE ESTADO DE LA VDP

Como puede verse en la tabla E.3, la CPU puede leer el contenido del registro de estado con una única transferencia de un byte. MODE estará a nivel alto para la transferencia. CSR se utiliza para indicar al VDP que se va a realizar una operación de lectura. Cada lectura del registro de estado dura, como mínimo, 2 microsegundos.

LECTURA POR LA CPU DE LA VRAM

La CPU lee la VRAM a través del VDP, utilizando el registro de direcciones de incremento automático. Una vez que la dirección ha sido definida todo lo que se precisa para leer el byte direccionado de la VRAM, es una transferencia de un byte. A continuación, el registro de direcciones se incrementa automáticamente. Por lo tanto, la lectura secuencial de datos de la VRAM, requiere una única transferencia de un byte, puesto que el registro de direcciones se encuentra ya preparado. Durante la carga del registro de direcciones, los dos bits más significativos del segundo byte de direcciones deben ser 0. Preparando, de esta forma, la dirección, se inicia un ciclo de lectura de la VRAM, y el dato leido estará disponible, para la primera transferencia a la CPU (veáse la tabla E.3). MODE estará a nivel alto para las transferencias de bytes de direcciones, y bajo, para las transferencias de datos.

La CPU se relaciona con la VRAM, a través del VDP. La duración de la preparación de dirección de escritura es de 4 microsegundos, y la de lectura, de 2 microsegundos. El tiempo total necesario para una transferencia de un byte de datos entre la CPU y la VRAM, puede variar desde 2 á 8 microsegundos. Una vez que se ha ordenado al VDP escribir o leer un byte de datos a la VRAM, se consumen aproximadamente 2 microsegundos hasta que el VDP está preparado para hacer la transferencia de datos. Este tiempo se mide desde el borde inicial de CSW del byte 3 para escritura, y del borde inicial de CSW del byte 2 para lectura. Además de este retraso de 2 microsegundos, el VDP debe esperar una ventana de acceso a CPU (es decir, el período de tiempo durante el cual el VDP no está ocupado con el refresco de memoria o visualizando pantalla, y en el que, por tanto, está disponible) para leer o escribir datos.

La duración, en el peor de los casos, del intervalo entre ventanas, se produce en el modo gráfico I ó II, utilizando sprites (veáse la tabla E.4). Durante la visualización activa, las ventanas a la CPU se producen una vez cada 16 ciclos de memoria, con un retraso máximo de 6 microsegundos (un ciclo de memoria dura unos 372 nanosegundos). En el modo de texto, las ventanas a la CPU, se producen al menos una vez cada tres ciclos de memoria, con un retraso, en el peor de los casos, de unos 1.1 microsegundos. Por último, en el modo multicolor las ventanas a la CPU se producen, al menos una vez, cada 4 ciclos de memoria.

Si el usuario necesita acceder a la memoria en 2 microsegundos, hay dos situaciones en las que el tiempo de espera para una ventana de acceso es 0. Ambas son independientes del modo de pantalla utilizado.

La primera situación de produce cuando el bit blank del registro 1 es 0. Con este bit bajo, la pantalla completa mostrará, únicamente, el color de borde, y el VDP no necesita esperar una ventana de acceso a CPU.

La segunda situación ocurre durante el intervalo vertical entre barridos. El VDP emite una salida de interrupción al final de cada área activa. Esta señal indica que el VDP está entrando en el retorno vertical entre barridos, y que durante los próximos 4.3 milisegundos, no es necesario esperar para una ventana de acceso. Si el usuario desea que la CPU acceda a la memoria durante este intervalo, ésta debe vigilar la salida de interrupción de la CPU (la CPU puede, bien esperar esta salida, o utilizarla como entrada de interrupción).

El programa que vigila la salida de interrupción, debe permitir sus propios retrasos de respuesta a la señal de interrupción, y conocer la duración del período de refresco de 4.3 milisegundos. La CPU debe escribir un 1 al bit de activación de interrupciones del registro 1, en la inicialización para activar la interrupción en cada cuadro. A continuación debe leer el registro de estado cada vez que se produzca una interrupción.

Tabla E.4 Tiempos de retardo en el acceso a memoria.

CONDICION	MODO	RETARDO DE VDP	TIEMPO DE ESPERA PARA UNA VENTANA DE ACCESO	TIEMPO TOTAL
Area áctiva de pantalla	Texto	2 us	0-1.1 us	2-3.1 us
Area activa de pantalla	Gráficos I,II	2 us	0-5.95 us	2-8 us
4300 us después de señal interrup. vert.	Todos	2 us	0 us	2 us
Bit blank registro 1 0	Todos	2 us	0 us	2 us
Area activa de pantalla	Multicolor	2 us	0-1.5 us	2-3.5 us

CPU CONTROLADA MEDIANTE INTERRUPCIONES

En un medio controlado por interrupciones (CPU que acepta interrupciones), una interrupción puede producirse antes de que cualquiera de las secuencias de la tabla E.3 haya terminado. Por ejemplo, se puede producir una interrupción inmediatamente después de haber cargado el byte 1 ó byte 2 de direcciones durante una operación de escritura a la VRAM. En este caso, la rutina de servicio de la interrupción desconoce en qué punto, dentro de la secuencia, se produjo la interrupción. Por lo tanto, es necesario desactivar y reactivar las interrupciones, antes y después, de cada secuencia de preparación. Esta acción previene la pérdida de continuidad entre la CPU y el VDP.

Si esta continuidad no es importante, puede leerse el registro de estado. Los circuitos internos de interfaz con la CPU se reinicializan y aceptarán el siguiente byte como el primer elemento de una nueva interconexión entre CPU y VDP. Si ninguna técnica es aceptable, se puede emplear un sondeo.

INTERRUPCION DEL VDP

La patilla de salida $\overline{INT}$ del VDP, genera una interrupción al final de cada barrido de la pantalla activa (aproximadamente, cada 1/60 seg. para el TMS 9118/9128, y 1/50 seg. para el TMS 9129). La salida $\overline{INT}$ estará activa cuando el bit de activación de interrupciones del registro 1 del VDP esté a nivel 1, y la bandera F de estado del registro de estado, se ponga también a 1. Las interrupciones se anulan cuando se lee el registro.

La interrupción del VDP se produce al final de la visualización del plano patrón activo, antes de mostrar las últimas líneas del fondo. Esta interrupción puede utilizarse para mover sprites y alterar el plano patrón, pero no es utilizable para cambiar rápidamente el color de fondo.

$\overline{RESET}$/SYNC

El VDP se inicializa externamente cuando la entrada $\overline{RESET}$/SYNC esté activa (nivel bajo) y se mantenga a nivel bajo, durante un mínimo de 3 microsegundos. La reposición externa sincroniza todos los relojes con su borde de caida, fija los contadores horizontal y vertical a estados conocidos, y anula los registros 0 y 1 del VDP. La visualización de video desaparece, puesto que el bit blank del registro 1 del VDP, se convierte en 0. Sin embargo, el VDP continúa refrescando la memoria a pesar de anularse la visualización. Para restaurar la figura, deben escribirse los valores correctos a los registros 0 y 1 del VDP. Mientras la línea $\overline{RESET}$/SYNC esté activa, el VDP no refresca la VRAM.

REGISTROS QUE SOLO ADMITEN ESCRITURA

Los 8 registros de escritura del VDP, se muestran en la figura E.4. Se cargan por la CPU, por el procedimiento ya descrito. Los registros 0 y 1 contienen banderas para activar o desactivar varios modos y funciones del VDP. Los registros 2 al 6 definen las direcciones de base de varios subbloques dentro de la VRAM. Estos subbloques forman tablas utilizadas para producir la imagen deseada en la pantalla. El contenido de estas tablas debe ser creado por el microprocesador. El registro 7 se utiliza para definir el color de fondo y de texto.

Cada registro se describe a continuación.

REGISTRO 0

El registro 0 contiene dos bits de control de opciones del VDP. Los restantes bits se reservan para un uso futuro, y deben ser 0.

Bit 6 M3 (bit de modo 3) Veáse el apartado siguiente con su descripción.

Bit 7 Activación/desactivación de plano de VDP externo. Si es 0, se desactiva el plano de VDP externo, si es 1, se activa.

Figura E.4 Registros del VDP

REGISTRO	MSB							LSB
0	0	0	0	0	0	0	M3	EV
1	dā igual	blank	IE	M1	M2	0	tamaño	ampliación
2	0	0	0	0	dirección base tabla nombres patrones			
3	dirección base tabla color patrones							
4	0	0	0	0	0	dirección base tabla patrones generadores		
5	0	dirección base tabla atributos sprite						
6	0	0	0	0	0	dirección base tabla patrones generadores sprite		
7	color texto 1				color texto 0/color fondo			
estado sólo lectura	F	5S	C	número quinto sprite				

REGISTRO 1 (CONTIENE 6 BITS DE CONTROL DE OPCIONES DE VDP)

Bit 0 Bit no utilizado.

Bit 1 Activación/desactivación BLANK.
El valor 0 anula la visualización de la pantalla, dejando sólamente el color de borde. El valor 1 activa la pantalla.

Bit 2 IE (Activación de interrupciones). El valor 0 desactiva la interrupción del VDP, el valor 1 la activa.

Bit 3,4 M1, M2 (Bits de modo 1 y 2). M1, M2 y M3 determinan el modo operativo del VDP:

M1	M2	M3	
0	0	0	Modo gráfico I
0	0	1	Modo gráfico II
0	1	0	Modo multicolor
1	0	0	Modo de texto

Bit 5 Reservado.

Bit 6 Selecciona el tamaño de sprite. El valor 0 selecciona el tamaño 0 de sprites (8*8). El valor 1 selecciona el tamaño 1 (16*16).

Bit 7 Opción de ampliación de sprites. El valor 0 selecciona la ampliación 0 (1*). El valor selecciona la ampliacion 1 (2*)

REGISTRO 2

El registro 2 define la dirección base del subbloque tabla de nombres de patrones. Esta tabla contiene el nombre o el puntero de una definición de patrón de la tabla de patrones generadores. El rango de este registro va de 0 á 15. El contenido del registro constituye los 4 bits superiores de las direcciones de 14 bits de la tabla de nombres de patrones, por lo tanto, la dirección base de la tabla de nombres de patrones es equivalente a (registro 2)*400H. La tabla E.5 muestra las posibles direcciones de comienzo de la tabla de nombres de patrones.

Tabla E.5 Direccionamiento mediante el Registro 2

R2 * 400H = Dirección de comienzo

R2	DIRECCION
00	0000
01	0400
02	0800
03	0C00 - número máximo para 4K RAM
04	1000
05	1400
06	1800
07	1C00
08	2000
09	2400
0A	2800
0B	2C00
0C	3000
0D	3400
0E	3800
0F	3C00 - número máximo

REGISTRO 3

El registro 3 define la dirección de base del subbloque tabla de color

de patrones. La tabla de color de patrones define el color de los unos y ceros que los forman. El rango del registro 3 va de 0 á 255. El contenido del registro constituye los 8 bits superiores de las direcciones de 14 bits de la tabla de color de patrones, por lo tanto, la dirección de base de la tabla de color de patrones es equivalente a (registro 3)*40H. El registro 3 opera en forma diferente cuando el VDP se encuentra en el modo gráfico II. En este modo, la tabla de color de patrones puede situarse solamente en uno de dos lugares de la VRAM, bien por encima de 0000, ó por encima de 2000. En el primer caso, el bit más significativo del registro 3, debe ser 0, en el segundo caso, debe ser 1. Los bits 1 á 7 del registro 3, deben ponerse a 1. Por lo tanto, en el modo gráfico II, los únicos dos valores que operan correctamente en el registro 3, sobrepasan 7F y FF.

REGISTRO 4

El registro 4 define la dirección de base del subbloque tabla de patrones generadores. La tabla de patrones generadores contiene una biblioteca de los patrones que pueden visualizarse en la pantalla. El rango del registro 0 está comprendido entre 0 y 7. El contenido del registro conforma los 3 bits superiores de las direcciones de 14 bits del generador. Por lo tanto, la dirección base de la tabla de patrones generadores es equivalente a (registro 4)*800H. La tabla E.6 muestra las posibles direcciones de comienzo del subbloque tabla de patrones generadores, con la excepción del modo gráfico II.

El registro 4 opera en forma diferente cuando el VDP está en el modo gráfico II. En este modo, la tabla de patrones generadores solamente puede situarse en uno de dos lugares de la VRAM, bien a partir de 0000 ó de 2000. En el primer caso, el bit 5 del registro 4, debe ser 0, y 1 en el segundo. En ambos casos los bits 6 y 7 del registro 4, deben ser 1. Por lo tanto, en el modo gráfico II, los únicos dos valores del registro 4 que operan correctamente, son los situados a partir de 03 y 07.

El bit 5 del registro 4 debe tener valor opuesto al bit 0 del registro 3 para que las dos zonas de 8K estén separadas. En caso contrario, el modo gráfico II no funcionará.

Tabla E.7 Direccionamiento mediante el Registro 4

R4 * 800H = Dirección de comienzo

R4	DIRECCION
00	0000
01	0800 - máximo para 4K RAM
02	1000
03	1800
04	2000
05	2800
06	3000
07	3800 - máximo para 16K RAM

REGISTRO 5

El registro 5 define la dirección de base del subbloque tabla de atributos de sprite. La tabla de atributos de sprite especifica dónde se mostrará el sprite en pantalla. El rango del registro 5 va de 0 á 127. El contenido del registro conforma los 7 bits superiores de la dirección de 14 bits de la tabla de atributos de sprite. Por lo tanto, la dirección base es igual a (registro 5)*80H.

REGISTRO 6

El registro 6 define la dirección de base del subbloque de patrones generadores de sprite. La tabla de patrones generadores de sprite describe la apariencia del sprite. El rango del registro 6 va de 0 á 7 El contenido del registro 6 constituye los 3 bits superiores de la dirección de 14 bits de patrones generadores. Por lo tanto, la dirección base de la tabla de patrones de sprite, es equivalente a (registro 6)*800H.

La tabla E.8 muestra las posibles direcciones de comienzo del subbloque de patrones generadores de sprite.

Tabla E.8 Direccionamiento mediante el registro 6

R6 * 800H = Dirección de comienzo

R6	DIRECCION
00	0000
01	0800 - máximo para 4K RAM
02	1000
03	1800
04	2000
05	2800
06	3000
07	3800 - máximo para 16K RAM

REGISTRO 7

Los 4 bits superiores del registro 7 contiene el código del color 1 en el modo de texto, y los 4 bits inferiores, el del color 0, equivalentes al color de fondo y de borde en los otros modos.

REGISTRO DE ESTADO

El VDP tiene un único registro de estado de 8 bits, que sólo admite lectura, al que puede acceder la CPU. El registro de estado contiene la bandera de interrupción (F), la bandera de coincidencia de sprites (C), la bandera de quinto sprite (5S) y el número del quinto sprite (si existe). El formato del registro de estado se muestra en la figura E 4.

El registro de estado puede leerse en cualquier momento, para comprobar las banderas F, C y 5S. La lectura de este registro borrará la bandera de interrupción F.

La lectura en modo asíncrono, pondrá a 0 la bandera de interrupción, perdiendo su contenido. Por lo tanto, el registro de estado debe leerse, solamente, cuando está pendiente la interrupción del VDP.

BANDERA DE INTERRUPCION (F)

La bandera de interrupción F del registro de estado, se pone a 1 al final del barrido de la última línea de la pantalla activa. Se pone a 0 después de la lectura del registro de estado, o cuando el VDP se repone externamente. Si el bit de activación de interrupciones (IE) del registro 1 del VDP está activo (1), la salida de interrupciones (INT) estará activa (nivel bajo), siempre que la bandera de interrupción valga 1.

El registro de estado debe leerse cuadro a cuadro, para eliminar la interrupción y recibir la nueva interrupción del siguiente cuadro.

BANDERA DE COINCIDENCIA DE SPRITE (C)

La bandera C de coincidencia de sprites del registro de estado adquiere el nivel 1, si 2 ó más sprites coinciden, la coincidencia se produce si, cualesquiera dos sprites de la pantalla, tienen al menos un punto superpuesto. Se consideran incluidos los sprites transparentes y tambien aquellos que estén, parcial o totalmente, fuera de la pantalla. Para convertir un sprite individual en invisible, es preciso seleccionar el color transparente. La bandera C se pone a 0 después de la lectura del registro de estado, o de la reposición externa del VDP. La puesta a nivel 1 de la bandera C no produce una interrupción.

Los sprites situados más allá del marcador de fin de la tabla de atributos de sprite (mayor D0) no se consideran. Por ejemplo, si se sitúa un valor mayor que 0 en el byte de posición vertical del sprite 3 de la tabla de atributos de sprite, los sprites desde el 4 al 31, se haran invisibles.

El registro de estado debe leerse inmediatamente después de la conexión del sistema, para asegurarse de que la bandera de coincidencia está a 0.

El VDP compueba cada posición de punto, buscando la posible coincidencia, durante la generación del propio punto, independientemente de dónde se sitúe sobre la pantalla. Este proceso se realiza cada 1/60 seg. para el TMS 9118 y 9128, y cada 1/50 seg. para el TMS 9129. Por lo tanto, cuando los sprites móviles se desplazan más de una coordenada durante estos intervalos, es posible que los sprites tengan varios puntos superpuestos, o incluso, haber pasado completamente, uno sobre otro, para cuando el VDP compruebe la coincidencia.

BANDERA DE QUINTO SPRITE (5S) Y SU NUMERO

La bandera 5S de quinto sprite del registro de estado se pone a 1 cuando hay cinco o más sprites en la misma línea horizontal (líneas 1 á 192), y la bandera de interrupción valga 0. La bandera de quinto sprite adquiere el nivel 1 incluso si los sprites están situados fuera de la pantalla. La bandera 5S se anula después de la lectura del registro de estado, o cuando el VDP se repone externamente. El número del quinto sprite se almacena en los 5 bits inferiores del registro de estado, cuando la bandera 5S se pone a 1, y es válido mientras la bandera

mantenga este valor. El nivel 1 de la bandera de quinto sprite, no produce una interrupción.

OSCILADOR Y GENERADOR DE TIEMPOS

El VDP está diseñado para funcionar con una entrada de cristal de frecuencia 10.738653 (+/-0.0005) MHz. para generar la señales de reloj internas. Se precisa un cristal que oscile, en su frecuencia fundamental, en modo paralelo, para el oscilador de reloj interno, que es la base de tiempos maestra para todas las operaciones del sistema. Este reloj maestro se divide por 2 para generar el reloj de puntos (5.3 MHz.) y por 3, para generar la señal CPUCLK (3.58 MHz. solamente para el TMS 9118).

Los cristales para el TMS 9118/9128/9129 pueden adquirirse de una de las siguientes compañías o de sus distribuidores autorizados:

NDK
10080 North Wolfe Road Suite 220
Cupertino, California 95014
Teléfono: 408-255-0831
Telex: 352057

CTS Knight Inc.
400 Reimann Avenue
Sandwich, Illinois 60548
Teléfono: 815-786-8411

APENDICE F

GI AY-3-8910 PSG

(Extracto del Manual del Generador Programable de Sonido AY-3-8910, reproducido con permiso de General Instruments Microelectronics Ltd.)

INTRODUCCION

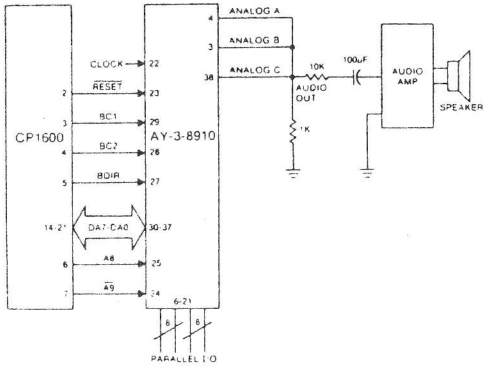
El generador de sonido programable AY-3-8910/8912 destaca por las siguientes características:

- Control absoluto mediante programa de la generación de sonido.
- Interconexión con la mayoría de los microprocesadores de 8 y 16 bits.
- 3 Salidas analógicas de programación independiente.
- 2 Puertos de E/S de 8 bits de uso general (AY-3-8910).
- 1 Puerto de E/S de 8 bits de uso general (AY-3-8912).
- Alimentación única de +5 voltios.

Este manual de datos pretende presentar las técnicas necesarias para hacer que el generador programable de sonido AY-3-8910/8912 realice las funciones para las que ha sido diseñado. Todos los programas, formas de programación y diseños de circuitos han sido comprobados para asegurar su utilidad practica más que puramente teórica.

Aunque las técnicas descritas producirán resultados espectaculares, el amplio espectro de sonidos a sintetizar es tan grande, y las posibilidades del PSG tan variadas, que esta guía debe verse como una introducción a las aplicaciones del PSG.

Figura F.1 Diagrama del sistema básico.



ASIGNACION DEL PATILLAS

El AY-3-8910 se suministra en un encapsulado de 40 patillas tipo D.I.L.P. con las asignaciones de patillas que se muestran en la figura F.2. El AY-3-8912 se suministra en encapsulado D.I.L.P. de 28 patillas con la asignación de la figura F.3

Figura F.2 Asignación de patillas del AY-3-8910

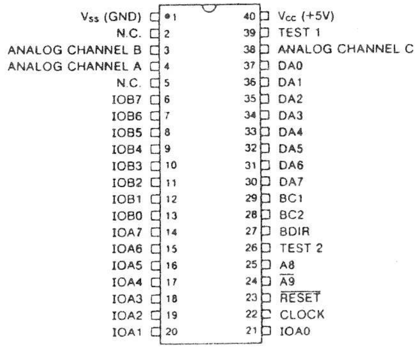
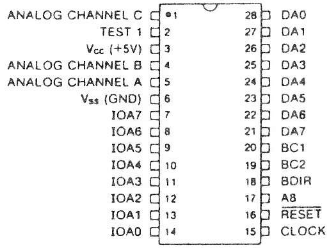


Figura F.3 Asignación de patillas del AY-3-8912



FUNCIONES DE LAS PATILLAS

DA7-DA0 (Entrada/Salida/Alta Impedancia): Patillas 30-37 (AY-3-8910)
Dirección de datos 7-0: 21-28 (AY-3-8912)

Estas ocho líneas comprenden el bus bidireccional de 8 bits, utilizado por el microprocesador para enviar tanto los datos como las direcciones al PSG, y recibir datos del PSG. En el modo de datos DA7-DA0, corresponden a los bits B7-B0 del registro conjunto. En el modo de direcciones, DA3-DA0 seleccionan el registro # $(0\text{-}17_8)$ y DA7-DA4, en conjunción con las entradas de dirección $\overline{A9}$ y A8, conforman la dirección de mayor orden (selección de circuito).

A8 (Entrada): Patilla 25 (AY-3-8910)
Patilla 17 (AY-3-8912)

$\overline{\text{A9}}$ (Entrada): Patilla 24 (AY-3-8910)
(Inexistente en el AY-3-8912)

$\overline{\text{Dirección 9}}$, Dirección 8

Estos bits de dirección extra, están disponibles para permitir el posicionamiento del PSG (asignando un espacio de memoria de 16 palabras) con un área total de memoria de 1024 palabras en vez de 256 como era definida por los bits de direcciones DA7-DA0 aislados. Si el tamaño de memoria no precisa de la utilización de estas líneas de direcciones extra, pueden dejarse de conectar, al estar provistas de una resistencia desconectable ($\overline{\text{A9}}$) ó conectable (A8). En medios ruidosos, sin embargo, es recomendable que $\overline{\text{A9}}$ y A8 se conecten a una tierra externa y a +5V, respectivamente, si no se utilizan.

$\overline{\text{RESET}}$ (Entrada): Patilla 23 (AY-3-8910)
Patilla 16 (AY-3-8912)

Utilizada para la inicialización/conexión del circuito. Aplicando un 0 lógico (tierra) a la patilla $\overline{\text{Reset}}$, se repondrán todos los registros a 0. La patilla $\overline{\text{Reset}}$ está provista de una resistencia en el propio circuito, desconectable.

CLOCK (Entrada): Patilla 22 (AY-3-8910)
Patilla 15 (AY-3-8912)

Esta entrada, compatible con lógica TTL, suministra la referencia de tiempos para los generadores de tono, ruido y envolvente.

BDIR, BC2, BC1 (Entradas): Patillas 27, 28, 29 (AY-3-8910)
Patillas 18, 19, 20 (AY-3-8912)

Bus DIRection, Bus Control 2,1

Estas señales de control de bus se generan directamente por la serie de microprocesadores de GI CP 1600, y controlan todas las operaciones externas e internas del bus del PSG. Cuando se utilice un procesador distinto de los CP 1600, estas señales deben sustituirse por otras señales de bus similares, o simulando las señales mediante las líneas de E/S del procesador. El PSG decodifica estas líneas como se muestra a continuación.

BDIR	BC2	BC1	FUNCION CP 1600	FUNCION PSG
0	0	0	NACT	Inactivo, véase 010 (IAB) debajo
0	0	1	ADAR	Dirección de enganche, véase 111 (INTAK) debajo.
0	1	0	IAB	Inactivo, el bus PSG/CPU está inactivo. DA7-DA0 están en estado de alta impedancia.
0	1	1	DTB	Lectura del PSG. Esta señal produce que el contenido del registro direccionado aparezca en el bus PSG/CPU. DA7-DA0 están en modo de salida.
1	0	0	BAR	Dirección de enganche, véase 111 (INTAK) debajo.
1	0	1	DW	Inactivo, véase 010 (IAB) encima.
1	1	0	DWS	Escritura al PSG. Esta señal indica que el bus contiene datos de registro que deben ser recibidos en el registro direccionado. DA7-DA0 en modo entrada.
1	1	1	INTAK	Dirección de enganche. Esta señal indica que el bus contiene una dirección de registro que debe introducirse en el PSG. DA7-DA0 en modo entrada.

Cuando la interconexión se realice con un procesador distinto del CP 1600, se deben simular las señales de codificación anteriores, la redundancia en las funciones del PSG en relación con las señales de control del bus, pueden utilizarse con la ventaja de que solamente 4 de las 8 posibles funciones codificadas en el bus, se precisan por el PSG. Se puede, así, simplificar la programación de las señales de bus de control, a las siguientes, que sólo requieren que el procesador genere dos señales de bus de control (BDIR y BC1, estando BC2 conectada a +5V):

BDIR	BC2	BC1	FUNCION PSG
0	1	0	Inactivo
0	1	1	Lectura del PSG
1	1	0	Escritura al PSG
1	1	1	Dirección de enganche

CANAL ANALOGICO A, B, C (Salidas): Patillas 4, 3, 38 (AY-3-8910)
Patillas 5, 4, 1 (AY-3-8912)

Cada una de estas tres señales es la salida de su correspondiente convertidor D/A, y produce una señal de hasta 1Vpp. representando la compleja forma de onda del sonido generado por el PSG.

IOA7-IOA0 (Entrada/Salida): Patillas 14-21 (AY-3-8910)
Patillas 7-14 (AY-3-8912)

IOB7-IOB0 (Entrada/Salida): Patillas 6-13 (AY-3-8910)
(Inexistente en el AY-3-8912)

ENTRADA/SALIDA A7-A0, B7-B0

Cada uno de estos dos puertos paralelos de Entradas/Salidas permiten disponer de 8 bits de datos en el bus PSG/CPU conectables a cualquier dispositivo externo mediante las patillas IOA ó IOB. Cada patilla está provista de una resistencia desconectable, de forma que en el modo de entrada todas las patillas se leerán a nivel alto. Por lo tanto, el método recomendado para la exploración de interruptores externos, sería el de conectar a tierra el bit de entrada.

TEST 1: Patilla 39 (AY-3-8910)
Patilla 2 (AY-3-8912)

TEST 2: Patilla 26 (AY-3-8910)
(Inexistente en el AY-3-8912)

Estas patillas se emplean para comprobación en fábrica, y deben dejarse sin conectar. No deben emplearse como puntos de unión.

Vcc: Patilla 40 (AY-3-8910)
Patilla 3 (AY-3-8912)

Alimentación de +5V nominales del PSG.

Vss: Patilla 1 (AY-3-8910)
Patilla 6 (AY-3-8912)

Patilla de tierra del PSG.

TEMPORIZACION DE SEÑALES DEL BUS DE CONTROL

Como las funciones del PSG son controladas por las órdenes del procesador del sistema, el bus común de datos/direcciones requiere definición de su función en cada momento particular. Esto se realiza por el procesador mediante el envío de señales de control a través del bus, que ya han sido definidas, y que determinan el estado del bus. El PSG decodifica estas señales para ejecutar la tarea solicitada.

La generación de estas señales de control, a través del bus, por el procesador, es similar a las que el procesador produce al interactuar con la RAM: (1) el procesador envía una dirección de memoria; y (2) el procesador bien da salida o entrada a los datos, a/de la memoria. La memoria, en este caso, es el conjunto de 16 registros de lectura/escritura, del PSG.

Las relaciones del tiempo en el envío de las señales de control, en relación a las señales de datos o direcciones, se recogen en el siguiente apartado.

Figura F.4 Conjunto de registros del PSG

Registro	bit	B7	B6	B5	B4	B3	B2	B1	B0
R0	Canal A Período de Tono	Ajuste fino 8 bit canal A							
R1						Ajuste grueso 4bit A			
R2	Canal B Período de Tono	Ajuste fino 8 bit canal B							
R3						Ajuste grueso 4bit B			
R4	Canal C Período de Tono	Ajuste fino 8 bit canal C							
R5						Ajuste grueso 4bit C			
R6	Período de Ruido				Control de Período 5bit				
R7	Activación	E/S		Ruido			Tono		
		IOB	IOA	C	B	A	C	B	A
R10	Canal A Amplitud				M	L3	L2	L1	L0
R11	Canal B Amplitud				M	L3	L2	L1	L0
R12	Canal C Amplitud				M	L3	L2	L1	L0
R13	Período Envolvente	Ajuste fino 8 bit Envolvente							
R14		Ajuste grueso 8 bit Envolvente							
R15	Forma/Ciclo Envolvente					CONT	ATQ	ALT	MANT
R16	Almacén datos Puerto A	E/S Paralelo 8 bit Puerto A							
R17	Almacén datos Puerto B	E/S Paralelo 8 bit Puerto B							

NOTA	OCTAVA	FRECUENCIA IDEAL	FRECUENCIA ACTUAL	REGISTRO 12BIT VALOR EN OCTAL			
C	1	32.703	32.698	6	5	3	5
C#	1	34.648	34.653	6	2	3	4
D	1	36.708	36.712	5	7	4	7
D#	1	38.891	38.895	5	4	7	4
E	1	41.203	41.201	5	2	3	3
F	1	43.654	43.662	5	0	0	2
F#	1	46.249	46.243	4	5	6	3
G	1	48.999	48.997	4	3	5	3
G#	1	51.913	51.908	4	1	5	3
A	1	55.000	54.995	3	7	6	2
A#	1	58.270	58.261	3	6	0	0
B	1	61.735	61.733	3	4	2	4
C	2	65.406	65.416	3	2	5	6
C#	2	69.296	69.307	3	1	1	6
D	2	73.416	73.399	2	7	6	4
D#	2	77.782	77.789	2	6	3	6
E	2	82.406	82.432	2	5	1	5
F	2	87.308	87.323	2	4	0	1
F#	2	92.498	92.523	2	2	7	1
G	2	97.998	98.037	2	1	6	5
G#	2	103.826	103.863	2	0	6	5
A	2	110.000	109.991	1	7	7	1
A#	2	116.540	116.522	1	7	0	0
B	2	123.470	123.467	1	6	1	2
C	3	130.812	130.831	1	5	2	7
C#	3	138.592	138.613	1	4	4	7
D	3	146.832	146.799	1	3	7	2
D#	3	155.564	155.578	1	3	1	7
E	3	164.812	164.743	1	2	4	7
F	3	174.616	174.510	1	2	0	1
F#	3	184.996	184.894	1	1	3	5
G	3	195.996	195.903	1	0	7	3
G#	3	207.652	207.534	1	0	3	3
A	3	220.000	220.198	0	7	7	4
A#	3	233.080	233.043	0	7	4	0
B	3	246.940	246.933	0	7	0	5
C	4	261.624	261.357	0	6	5	4
C#	4	277.184	276.883	0	6	2	4
D	4	293.664	293.598	0	5	7	5
D#	4	311.128	310.724	0	5	5	0
E	4	329.624	329.973	0	5	2	3
F	4	349.232	349.565	0	5	0	0
F#	4	369.992	370.400	0	4	5	6
G	4	391.992	392.494	0	4	3	5
G#	4	415.304	415.839	0	4	1	5
A	4	440.000	440.397	0	3	7	6
A#	4	466.160	466.087	0	3	6	0
B	4	493.880	494.959	0	3	4	2

NOTA	OCTAVA	FRECUENCIA IDEAL	FRECUENCIA ACTUAL	REGISTRO 8 BIT VALOR EN OCTAL			
C	6	523.248	522.714	0	3	2	6
C#	5	554.368	553.766	0	3	1	2
D	5	587.328	588.741	0	2	7	5
D#	5	622.256	621.449	0	2	6	4
E	5	659.248	658.005	0	2	5	2
F	5	698.464	699.130	0	2	4	0
F#	5	739.984	740.800	0	2	2	7
G	5	783.984	782.243	0	2	1	7
G#	5	830.608	828.598	0	2	0	7
A	5	880.000	880.794	0	1	7	7
A#	5	932.320	932.173	0	1	7	0
B	5	987.760	989.918	0	1	6	1
C	6	1046.496	1045.428	0	1	5	3
C#	6	1108.736	1107.532	0	1	4	5
D	6	1174.656	1177.482	0	1	3	7
D#	6	1244.512	1242.898	0	1	3	2
E	6	1318.496	1316.009	0	1	2	5
F	6	1396.928	1398.260	0	1	2	0
F#	6	1479.968	1471.852	0	1	1	4
G	6	1567.968	1575.504	0	1	0	7
G#	6	1661.216	1669.564	0	1	0	3
A	6	1760.000	1747.825	0	1	0	0
A#	6	1864.640	1864.346	0	0	7	4
B	6	1975.520	1962.470	0	0	7	1
C	7	2092.992	2110.581	0	0	6	5
C#	7	2217.472	2237.216	0	0	6	2
D	7	2349.312	2330.433	0	0	6	0
D#	7	2489.024	2485.795	0	0	5	5
E	7	2636.992	2663.352	0	0	5	2
F	7	2793.856	2796.520	0	0	5	0
F#	7	2959.936	2943.705	0	0	4	6
G	7	3135.936	3107.244	0	0	4	4
G#	7	3322.432	3290.023	0	0	4	2
A	7	3520.000	3495.649	0	0	4	0
A#	7	3729.280	3728.693	0	0	3	6
B	7	3951.040	3995.028	0	0	3	4
C	8	4185.984	4142.992	0	0	3	3
C#	8	4434.944	4474.431	0	0	3	1
D	8	4698.624	4660.866	0	0	3	0
D#	8	4978.048	5084.581	0	0	2	6
E	8	5273.984	5326.704	0	0	2	5
F	8	5587.712	5593.039	0	0	2	4
F#	8	5919.872	5887.410	0	0	2	3
G	8	6271.872	6214.488	0	0	2	2
G#	8	6644.864	6580.046	0	0	2	1
A	8	7040.000	6991.299	0	0	2	0
A#	8	7458.560	7457.385	0	0	1	7
B	8	7902.080	7990.056	0	0	1	6

Figura F.5 Escala cromatica uniformemente templada (f.reloj=1.78977 MHz) (C=do, D=re, E=mi ...)

FUNCIONAMIENTO

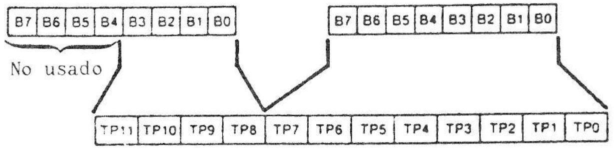
Como todas las funciones del PSG se controlan por el procesador maestro, mediante la carga de una serie de registros, la descripción detallada del funcionamiento del PSG, se reliza mejor relacionando cada función del mismo con el correspondiente registro. La creación o programación de un sonido específico o de un efecto de sonido, sigue la siguiente secuencia de control:

OPERACION	REGISTROS	FUNCION
Control generador tono	R0-R5	Programación de períodos de tono
Control generador ruido	R6	Programación de período de ruido
Control mezclador	R7	Activación tono y/o ruido en canales seleccionados
Control Amplitud	R10-R12	Selección de amplitudes fijas o variables por envolvente
Control generador envolvente	R13-R15	Programación de período envolvente y selección patrón envolvente

CONTROL DE GENERADOR DE TONO (REGISTROS R0-R5)

La frecuencia de cada onda cuadrada producida por los tres generadores de tono (uno por cada canal A, B y C) se obtiene, en el PSG, contando hacia atrás la entrada de reloj de 16 en 16, y después, con el valor del período de tono de 12 bits programado. Cada valor de 12 bits se obtiene por el PSG combinando los contenidos de los registros de ajuste fino y grueso, como se muestra a continuación:

REGISTRO AJUSTE GRUESO	CANAL	REGISTRO AJUSTE FINO
R1	A	R0
R3	B	R2
R5	C	R4



Período de tono de 12 bits (TP) al Generador de tono

Hay que destacar que el valor de 12 bits programado por la combinación de los registros de ajuste fino y grueso, es un valor de período -a mayor valor en los registros, menor frecuencia de tono resultante-.

Asímismo, y debido a la técnica de diseño utilizada en la cuenta atrás del período de tono, el valor de período menor es 000000000001 (divide por 1) y el periodo mayor es 111111111111 (divide por 4095_{10}).

Las ecuaciones que describen la relación entre la frecuencia del tono de salida deseado y la frecuencia de entrada de reloj y el valor de período de tono son:

(a) $f_T = \dfrac{f_{reloj}}{16TP_{10}}$ (b) $TP_{10} = 256CT_{10} + FT_{10}$

donde: f_T = frecuencia de tono deseado
f_{reloj} = frecuencia de entrada de reloj
TP_{10} = equivalente decimal de los bits de período de tono TP11-TP0
CT_{10} = equivalente decimal de los bits del registro de ajuste grueso B3-B0 (TP11-TP8)
FT_{10} = equivalente decimal de los bits del registro de ajuste fino B7-B0 (TP7-TP0)

De las ecuaciones anteriores se deduce que la frecuencia de tono está comprendida entre un mínimo de f.reloj/65520 (donde TP10=4095_{10}) hasta un máximo de f.reloj/16 (donde TP10=1) Utilizando una entrada de reloj de 2 MHz, por ejemplo, se obtendría un espectro de frecuencias de tono desde 30.5 Hz hasta 125 KHz

Para calcular los valores de los registros de ajuste de periodo de tono fino y grueso, dadas la frecuencia de entrada de reloj y la frecuencia de tono deseada, operamos en las anteriores ecuaciones obteniendo:

(a) $TP_{10} = \dfrac{f_{reloj}}{16f_T}$ (b) $CT_{10} + \dfrac{FT_{10}}{256} = \dfrac{TP_{10}}{256}$

Ejemplo 1: f_T = 1kHz; f_{reloj} = 2MHz

$TP_{10} = \dfrac{2x10^6}{16(1x10^3)} = 125;$ $CT_{10} + \dfrac{FT_{10}}{256} = \dfrac{125}{256}$

$CT_{10} = 0$ = 0000 (B3-B0)
$FT_{10} = 125_{10}$ = 01111101 (B7-B0)

Ejemplo 2: f_T = 100Hz; f_{reloj} = 2MHz

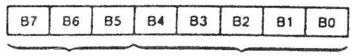
$TP_{10} = \dfrac{2x10^6}{16(1x10^2)} = 1250;$ $CT_{10} + \dfrac{FT_{10}}{256} = 4 + \dfrac{226}{256}$

$CT_{10} = 4_{10}$ = 0100 (B3-B0)
$FT_{10} = 226_{10}$ = 11100010 (B7-B0)

CONTROL DE GENERADOR DE RUIDO (REGISTRO R6)

La frecuencia de la fuente de ruido se obtiene, en el PSG, contando hacia atrás la entrada de reloj de 16 en 16, y después, con el valor del período de ruido de 5 bits programado. Este valor de 5 bits se obtiene de los 5 bits inferiores (B4-B0) del registro R6, como se muestra a continuación:

Período de ruido
Registro R6



No usado — Período de ruido de 5 bits (NP) al generador de ruido

El valor de 5 bits de R11 es un valor de período -a mayor valor en el registro, menor frecuencia de ruido resultante-. Al igual que con el período de tono, el valor inferior de período es 00001 (divide por 1) y el período mayor es 11111 (divide por 31_{10}). La ecuación de frecuencia de ruido es:

$$f_N = \frac{f_{reloj}}{16\ NP_{10}}$$

donde: f_N = frecuencia de ruido deseado
f_{reloj} = frecuencia de entrada de reloj
NP_{10} = equivalente decimal de los bits del registro de período de ruido B4-B0

De la ecuación anterior se deduce que la frecuencia de ruido está comprendida entre un mínimo de f.reloj/496 (donde NP10=31_{10}) hasta un máximo de f.reloj/16 (donde NP10=1). Utilizando una entrada de reloj de 2 MHz, por ejemplo, se obtendría un espectro de frecuencias de ruido, desde 4 KHz a 125 KHz.

Para calcular el valor del registro de período de ruido, dadas la frecuencia de entrada de reloj y la frecuencia de ruido deseada, operamos en la anterior ecuación obteniendo:

$$NP_{10} = \frac{f_{reloj}}{16 f_N}$$

CONTROL DE MEZCLADOR Y ACTIVACION (REGISTRO R7)

El registro R7 es un registro de activación multifuncional, que controla los tres mezcladores de ruido/tono y dos puertos de E/S de uso general.

Los mezcladores, como previamente se ha descrito, combinan las frecuencias de ruido y tono en cada uno de los tres canales. La determinación de esta combinación de ruido y/o tono en cada canal, se realiza por el estado de los bits B5-B0 de R7.

La actividad (entrada o salida) de los dos puertos de E/S de uso general (IOA, IOB) se determina por el estado de los bits B7 y B6 de R7.

Estas funciones se ilustran mediante el siguiente esquema:

Control de mezclador y activación
Registro R7

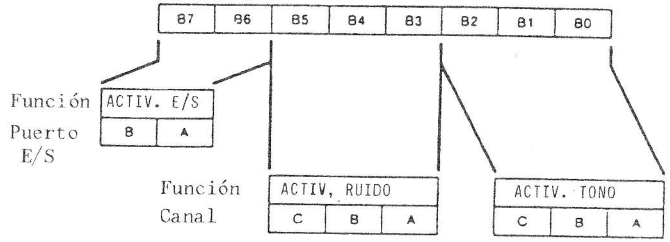


Tabla verdad Activación ruido:

R7 Bits B5	B4	B3	Ruido Activo en canal		
0	0	0	C	B	A
0	0	1	C	B	-
0	1	0	C	-	A
0	1	1	C	-	-
1	0	0	-	B	A
1	0	1	-	B	-
1	1	0	-	-	A
1	1	1	-	-	-

Tabla verdad Activación tono:

R7 Bits B2	B1	B0	Tono activo en canal		
0	0	0	C	B	A
0	0	1	C	B	-
0	1	0	C	-	A
0	1	1	C	-	-
1	0	0	-	B	A
1	0	1	-	B	-
1	1	0	-	-	A
1	1	1	-	-	-

Tabla verdad Puerto E/S:

R7 Bits B7	B6	Estado Puerto E/S IOB	IOA
0	0	entrada	entrada
0	1	entrada	salida
1	0	salida	entrada
1	1	salida	salida

La desactivación de ruido y tono no desconecta un canal. Para apagar un canal, se deben escribir ceros en el correspondiente registro de control de amplitud.

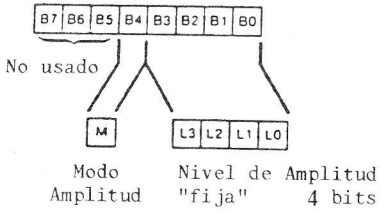
CONTROL DE AMPLITUD (REGISTROS R10, R11, R12)

Las amplitudes de las señales generadas por cada uno de los tres convertidores D/A (uno por canal A, B y C), se determinan por los contenidos de los 5 bits inferiores (B4-B0) de los registros R10, R11 y R12, como se muestra en el diagrama de la página siguiente.

El bit de modo de amplitud (bit M) selecciona, tanto la amplitud de nivel fijo (M=0), como la amplitud de nivel variable (M=1). Por lo tanto, los bits L3-L0 que definen el valor del nivel fijo de amplitud solamente están activos cuando M=0. Cuando se selecciona la amplitud de nivel fijo, ésta queda fijada únicamente en el sentido de que el nivel de amplitud está bajo el control directo del procesador del sistema (a través de los bits D3-D0). La variación de la amplitud en

Control de Amplitud
Registro # | Canal

Registro #	Canal
R10	A
R11	B
R12	C



el modo de amplitud fija, precisa en cada caso, de la intervención directa del procesador del sistema, a través de una secuencia de enganche y escritura de dirección/dato, para modificar el contenido de D3-D0.

Cuando M=1 (selección de amplitud con nivel variable), la amplitud de cada canal se determina por el patrón de envolvente definido por la salida de 4 bits del generador de envolvente (E0-E3).

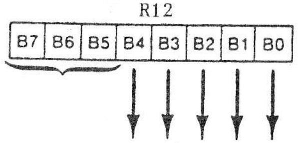
El bit M de modo de amplitud, puede verse también como un bit de activación de envolvente, esto es, cuando M=0, la envolvente no se utiliza, y cuando M=1, la envolvente está activada. (Una descripción completa del funcionamiento del generador de envolvente, puede verse en la sección siguiente).

La descripción de todas las combinaciones del control de amplitud de 5 bits, es la siguiente:

Control de Amplitud Registro #	Canal
R10	A
R11	B
R12	C

X= no importa su valor

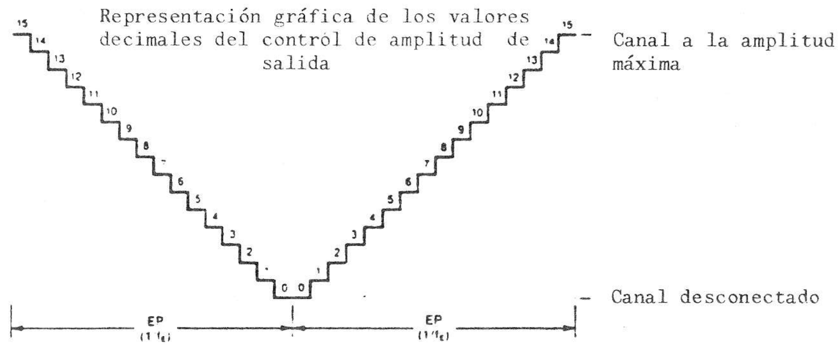
* el código formado por ceros apaga el canal



M	L3	L2	L1	L0	Control de Amplitud de salida	
0	0	0	0	0	*0 0 0 0	La amplitud está fijada en 1 de 16 niveles determinado por L3 L2 L1 L0.
.	.	.	.	.	. . . .	
.	.	.	.	.	. . . .	
0	1	1	1	1	1 1 1 1	
1	X	X	X	X	E3 E2 E1 E0	La amplitud varía entre 16 niveles según la salida del generador de envolvente.

La figura F 6 nos muestra el modo de amplitud variable (controlado por envolvente) en el que los 16 niveles reflejan la salida del generador de envolvente. Una amplitud de nivel fijo correspondería, solamente, a uno de los niveles mostrados, que dependería del equivalente decimal de los bits L3 L2 L1 y L0.

Figura F.6 Control de amplitud variable (M=1)

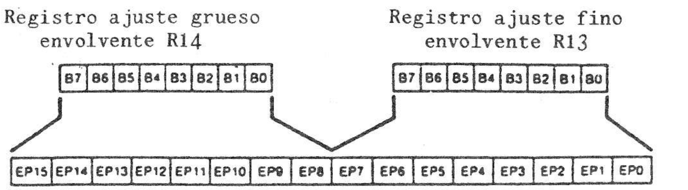


CONTROL DEL GENERADOR DE ENVOLVENTE (REGISTROS R13-R15)

Para permitir la generación de patrones de envolvente complejos se dispone de dos métodos independientes de control en el PSG. Por el primero, es posible variar la frecuencia de envolvente utilizando los registros R13 y R14. Por el segundo, se puede variar la forma y ciclo del patrón de envolvente, mediante el registro R15. Los siguientes párrafos explican los detalles de las funciones de control de envolvente, describiendo, en primer lugar, el control de período de envolvente, y después, el control de la forma/ciclo de envolvente.

CONTROL DE PERIODO DE ENVOLVENTE (REGISTROS R13-R14)

La frecuencia de envolvente se obtiene, en el PSG, por la cuenta atrás de 256 en 256 de la entrada del reloj, y después, con el valor de período de envolvente de 16 bits programado. Este valor de 16 bits se obtiene combinando el contenido de los registros de ajuste fino y grueso de envolvente, como se muestra en el siguiente diagrama:



Período envolvente 16 bits (EP) al generador de envolvente

E. ,alor de 16 bits programado por la combinación .. registros de ajuste fino y grueso es un valor de período -a mayor valor de los registros, menor será la frecuencia de envolvente resultante-.

Al igual que con el período de tono, el valor de período más bajo es 0000000000000001 (divide por 1), y el valor de período mayor es 1111111111111111 (divide por 65535_{10}).

Las ecuaciones de frecuencia de envolvente son:

$$(a)\quad f_E = \frac{f_{reloj}}{256EP_{10}} \qquad (b)\quad EP_{10} = 256CT_{10} + FT_{10}$$

donde: f_E = frecuencia de envolvente deseada

f_{reloj} = frecuencia de entrada de reloj

EP_{10} = equivalente decimal de los bits EP15-EP0 de período de envolvente

CT_{10} = equivalente decimal de los bits B7-B0 (EP15-EP8) del re gistro de ajuste grueso

FT_{10} = equivalente decimal de los bits B7-B0 (EP7-EP0) del re- gistro de ajuste fino

De la ecuación anterior puede deducirse que la frecuencia de envolvente está comprendida entre un mínimo de $f.reloj/16776960_{10}$ (donde: EP10=65535_{10}) hasta un máximo de f.reloj/256 (donde EP10=1). Si se utiliza un reloj de 2 MHz, por ejemplo, se obtendría un rango de frecuencias de envolvente entre 0.12 Hz y 7812.5 Hz.

Para calcular los valores de los registros de ajuste fino y grueso del período de envolvente, dadas las frecuencias de entrada de reloj y envolvente, operamos en las ecuaciones anteriores, obteniendo:

$$(a)\quad EP_{10} = \frac{f_{reloj}}{256f_E} \qquad (b)\quad CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$$

Ejemplo: $f_E = 0.5$ Hz; $f_{reloj} = 2$ Hz

$$EP_{10} = \frac{2x10^6}{256(0.5)} = 15625$$

Sustituyendo este resultado en la ecuación (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{15625}{256} = 61 + \frac{9}{256}$$

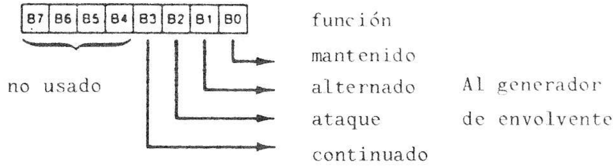
$CT_{10} = 61_{10} = 00111101$ (B7-B0); $FT_{10} = 9_{10} = 00001001$ (B7-B0)

CONTROL DE FO..A/CICLO DE ENVOLVENTE (REGISTRO R15)

El generador de envolvente cuenta hacia atrás la frecuencia de envolvente de 16 en 16, produciendo un patrón de envolvente de 16 estados por ciclo, según la definición dada por los 4 bits del contador de salida, E3 E2 E1 E0. La forma y ciclo de una envolvente se consigue mediante el control del patrón contador (cuenta adelante, cuenta atrás), del contador de 4 bits y definiendo un patrón de ciclo simple o de ciclo con repetición.

El control de la forma/ciclo de envolvente se encuentra en los 4 bits inferiores (B3-B0) del registro R15 Cada uno de estos 4 bits, controla una función del generador de envolvente, como se muestra en el diagrama siguiente:

Control de forma/ciclo
de envolvente Registro R15



La definición de cada función es la siguiente:

Mantenido: Cuando se fija a nivel lógico 1, limita la envolvente a 1 ciclo, manteniendo la última cuenta del contador de envolvente (E3-E0=0000 ó 1111, dependiendo de que el contador de envolvente estuviera en cuenta hacia delante o hacia atrás, respectivamente).

Alternado: Cuando se fija a nivel lógico 1, el contador de envolvente invierte la dirección de cuenta (adelante-atrás), después de cada ciclo. Cuando los 2 bits son unos, el contador de envolvente se repone a su cuenta inicial.

Ataque: Cuando se fija a nivel lógico 1, el contador de envolvente contará hacia delante (ataque), desde E3 E2 E1 E0=0000 hasta 1111; cuando se fija a nivel 0, el contador de envolvente contará hacia atrás (disminución) desde 1111 á 0000.

Continuado: Cuando se pone a nivel 1, el patrón de ciclo será el definido por el bit de mantenido; cuando se fija a nivel 0, el generador de envolvente se repone a 0000 después de 1 ciclo, y se mantiene en dicha cuenta.

Para describir más detalladamente las funciones anteriores, se podrían utilizar numerosas tablas de la secuencia de cuenta binaria de E3 E2 E1 E0 para cada combinación de mantenido, alternado, ataque y continuado. Sin embargo, como estas salidas se utilizan (cuando son seleccionadas por los registros de control de amplitud) para modular en amplitud la salida de los mezcladores, se consigue una mejor comprensión de su efecto a través de una representación gráfica de su valor en cada condición seleccionada, como es el caso de las figuras F.7 y F.8.

Figura F.7 Control de la forma/ciclo de envolvente

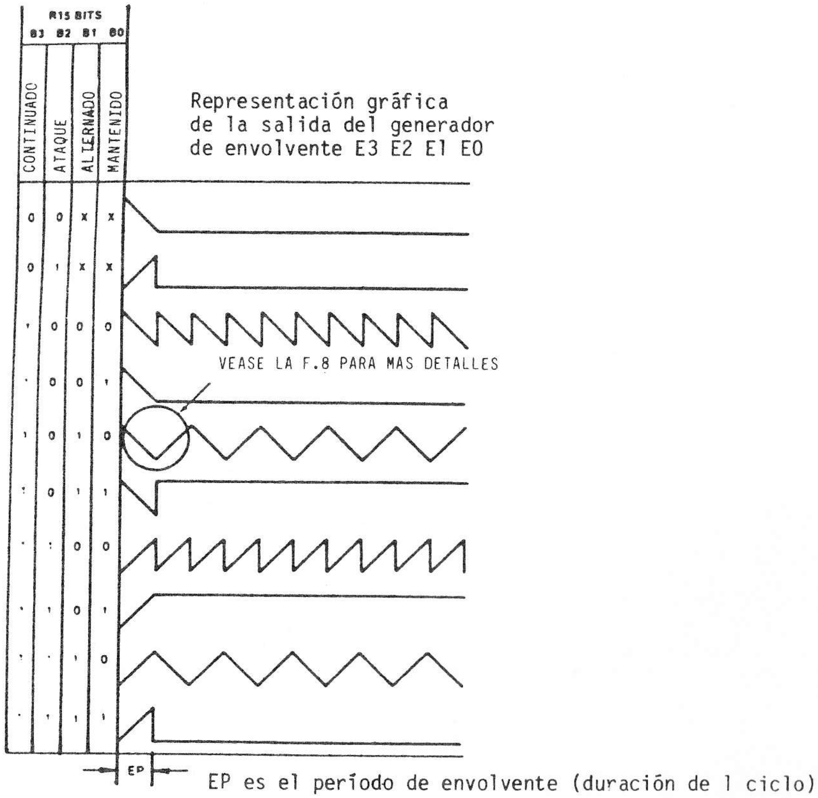
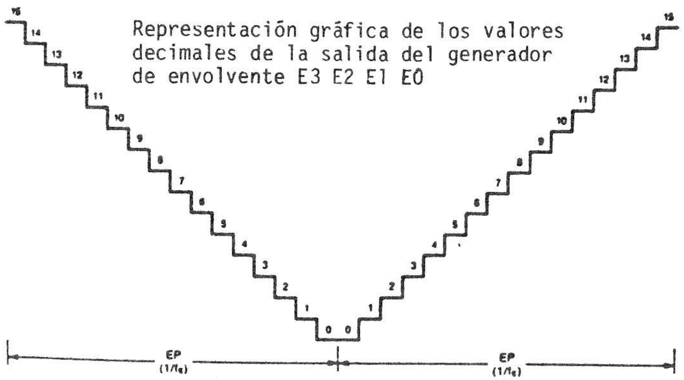


Figura F.8 Detalle de 2 ciclos de la figura F.7 (Forma de onda 1010)



GENERACION DE EFECTOS DE SONIDO

Uno de los usos principales del PSG es la producción de sonidos no musicales para acompañar la acción visual, o como una atracción en sí mismos. Las siguientes secciones desarrollan las técnicas y proporcionan ejemplos para la generación de algunos efectos conocidos. Todos los ejemplos están basados en un reloj de PSG de 1.78977 MHz.

EFECTOS DE TONO

Se pueden conseguir muchos efectos utilizando solamente la capacidad de generación de tono del PSG, sin añadir ruido y sin utilizar la generación de envolvente. Los ejemplos de este tipo de efecto incluyen frecuencias de tono de teléfono (2 frecuencias diferentes producidas simultáneamente) o el efecto de sirena europea (2 frecuencias diferentes producidas secuencialmente).

Sonido de sirena europea:

Registro #	Valor de carga octal	Explicación
otros	000	-
R0	376	Fija el período de tono del canal A
R1	000	a 2.27 ms (440 Hz)
R7	076	Activa tono canal A solamente
R10	017	Amplitud máxima en canal A
	(espera de 350 ms antes de continuar)	
R0	126	Fija el período de tono del canal A
R1	001	5346 ms (187 Hz)
	(espera de 350 ms antes de continuar)	
R10	000	Apaga canal A, fin efecto

EFECTOS DE RUIDO

Algunos de los sonidos más comunmente requeridos precisan, solamente, del uso de ruido y del generador de envolvente (o del control por procesador del canal de envolvente, si otros canales están utilizando el generador de envolvente).

Ejemplos de ello son: el disparo de bala y la explosión. En ambos casos se utiliza ruido puro con una envolvente en disminución. En los ejemplos mostrados, los únicos cambios son los de longitud de envolvente modificada por el registro de ajuste grueso, y el período de ruido. Puede obtenerse una explosión menor si los tres canales operan con los mismos parámetros.

Sonido de disparo de bala:

Registro #	Valor de carga octal	Explicación
otros	000	-
R6	017	Fija período de ruido a valor medio
R7	007	Activa ruido solamente en A, B, C
R10	020	Selecciona rango de amplitud total
R11	020	bajo control generador envolvente
R12	020	Fija período envolvente a 0.586 seg.
R14	020	Selecciona envolvente en disminución
R15	000	

Sonido de explosión:

Registro #	Valor de carga octal	Explicación
otros	000	-
R6	000	Fija período de ruido a valor máximo
R7	007	Activa ruido solamente en A, B, C
R10	020	Selecciona rango de amplitud total bajo control generador envolvente
R11	020	
R12	020	
R14	070	Fija período envolvente a 2.05 seg.
R15	000	Selecciona envolvente en disminución

EFECTOS DE BARRIDO DE FRECUENCIA

Los sonidos de laser, bomba silbante, aullido de lobo y coche de carreras, utilizan efectos de barrido de frecuencia. En estos casos, se emplean aumentos o disminuciones en los valores de los registros de período, con principio, fin y tiempo entre cambios de frecuencia, variables. Por ejemplo, la velocidad de barrido del laser es muy superior a la marcha más alta del coche de carreras, aunque ambos usen la misma rutina de ordenador con parámetros diferentes.

Otros efectos fáciles de conseguir son los de barrido de ruido y efecto "doppler". El barrido del registro de temporización de ruido (R6) produce un efecto "doppler" muy adecuado para juegos del tipo "guerra de las galaxias".

Sonido de laser:

Registro #	Valor de carga octal	Explicación
otros	000	-
R7	076	Activa tono solamente en canal A
R10	017	Selecciona amplitud máxima en canal A
R0	060(principio)	Efecto de barrido de período de tono canal A mediante bucle de procesador con tiempo de espera entre cada paso de 3 ms. variando entre 60 y 160 (0.429 ms/2330 Hz á 1.0 ms/1000 Hz)
R0	160(fin)	
R10	000	Apaga canal A

Sonido de bomba silbante:

Registro #	Valor de carga octal	Explicación
otros	000	
R7	076	Activa tono solamente en canal A
R10	017	Selecciona amplitud máxima en canal A
R0	060(principio)	Efecto de barrido de período de tono canal A mediante bucle de procesador con tiempo de espera entre cada paso de 25 ms. variando entre 60 y 300 (0.429 ms/2330 Hz á 1.72 ms/582 Hz)
R0	300(fin)	

EFECTOS MULTICANAL

Debido a la arquitectura independiente del PSG, se pueden generar muchos efectos bastante complejos sin que sea una tarea gravosa para el procesador. Por ejemplo, el aullido de lobo utiliza dos canales para añadir ruido de respiración silbante a los tres barridos de frecuencia del aullido. Una vez que el ruido ha entrado en el canal, el procesador solamente debe preocuparse de la operación de barrido de frecuencia.

Sonido de aullido de lobo:

Registro #	Valor de carga octal	Explicación
otros	000	-
R6	001	Fija período de ruido a valor mínimo
R7	056	Activa tono canal A y ruido canal B
R10	017	Selecciona amplitud máxima en canal A
R11	011	Selecciona amplitud mínima en canal B
R0	100 (principio)	Barrido período tono canal A mediante bucle de procesador con espera de 12 ms. entre pasos variando de 100 á 40 (0.572 ms/1748 Hz á 0.286 ms/3496Hz)
R0	040 (fin)	
(espera aproximadamente 150 ms. antes de continuar)		
R0	100 (principio)	+ bucle de procesador con espera de 25 ms. entre pasos variando de 100 á 60 (0.572 ms/1748Hz á 0.429ms/2331Hz)
R0	060 (fin)	
R0	060 (principio)	+ bucle de procesador con espera de 6 ms. entre pasos variando de 60 á 150 (0.429ms/2331Hz á 0.930ms/1075Hz)
R0	150 (fin)	
R10	000	Apaga canales A y B
R11	000	

Sonido de coche de carreras:

Registro #	Valor de carga octal	Explicación
otros	000	-
R3	017	Fija período canal B a 34.33ms(29Hz)
R7	074	Activa tono solamente en canales A, B
R10	017	Selecciona amplitud máxima en canal A
R11	012	Selecciona amplitud menor en canal B
*R1/R0	013/000 (principio)	Barrido período tono canal A mediante bucle de procesador con espera de 3 ms. entre pasos variando de 013/000 á 004/000 (25.17ms/39.7Hz á 9.15ms/109.3Hz)
*R1/R0	004/000 (fin)	
R1/R0	011/000 (principio)	+ bucle de procesador con espera de 3 ms. entre pasos variando de 011 / 000 á 003/000 (20.6ms/48.5Hz á 6.87 ms/145.6Hz)
R1/R0	003/000 (fin)	
R1/R0	006/000 (principio)	+ bucle de procesador con espera de 6 ms. entre pasos variando de 006 / 000 á 001/000 (13.73ms/72.8Hz á 2.29 ms/436.7Hz)
R1/R0	001/000 (fin)	
R10	000	Apaga canales A y B
R11	000	

INDICE

ABS 47
Acceso al VDP 100
Acceso rápido al VDP 148
Almacenamiento de programa 44
Almacenamiento de variables 46
AND 80
Archivos de programa 12
Archivos de programa en ASCII 12
Arquitectura del Z-80 80
ASC 47
ATN 47
AUTO 47

BASE 48
BEEP 42,48
BIN$ 48
Bit previo de reloj 35
BLOAD 48
BSAVE 49
Bus de datos 74
Bus de direcciones 74

CALL 49
Canal musical 155
Caracteres en la pantalla gráfica 38
Caracteres gráficos 29
CDBL 49
CHR$ 49
CINT 50
CIRCLE 40,50
Circuitos de sonido 42
CLEAR 24,37,50
CLOAD 51
CLOAD? 51
CLOSE 51
CLS 29,51
COLOR 18,27,51
Columnas reservadas 27
Configuración MSX 99
Conjuntos 54
Conmutación por interrupciones (tablas de sprites) 147
CONT 52
Control de motor de cassette 61
Control de velocidad de transferencia 67
Coordenadas de sprite 34
COS 52
CP 86
CPL 86
CSAVE 52
CSNG 52
CSRLIN 52

DATA 52
DEF FN 53
DEFDBL 53
Definición de sprite 35
DEFINT 53
DEFSNG 53
DEFSTR 53
DEFUSR 53
DELETE 53
Deteccion de interrupciones 61
Detección de colisión de sprites 37
DIM 53
Dimensiones 54
DRAW 54

Eco de teclado 67
Ejecución automática de programas 48
END 54
Entrada de mandos de juegos 69
EOF 54
ERASE 54
ERL 54
ERR 54
ERROR 54
E/S de juegos 161
E/S de teclado y pantalla 162
Espacio de cadenas 50
Estado de raquetas 63
Estado de tablero digitalizador 63
EXP 55
Exploración del teclado 164

FIX 55
FOR...NEXT 55
FRE 55
Funciones 25

Ganchos 102
Generación de números aleatorios 62
Generador Programable de sonido 9
Gestión de memoria 99
GOSUB 56
GOTO 56
Graficos Alta Resolución 18

HALT 92
HEX$ 56
IF.. THEN 56
INKEY$ 57
INP 56
INPUT 56
INPUT$ 56
INSTR 57
Instrucciones aritméticas del Z-80 85
Instrucciones de control de la CPU 91
Instrucciones del Z-80 82

Instrucciones Gráficas 26
INT 57
Interfaz Programable para periféricos 10
Interfaz de cassette 12
Interrupción modo 0, 1 94
Interrupción modo 2 95
INTERVAL ON 57
INTERVAL ON/OFF/STOP 57

KEY LIST 58
KEY ON/OFF 58

LEFT$ 58
LEN 58
Lenguaje máquina del Z-80 74
LINE 39,58
LINE INPUT 59
Lineas de control del Z-80 75
LIST 59
Llamadas a código máquina 70
LLIST 59
LOAD 59
LOCATE 27,59
LOG 60

Mandos de juegos 161
Manejo de errores 61
Manejo de interrupciones 102
Manipulación de la RAM de video 29
Mapa de bits 143
MAXFILES 60
Mensajes de error 54
MERGE 60
MID$ 60
Modos de direccionamiento del Z-80 96
Modos de pantalla 13
Modos de pantalla del VDP 116
Modos gráficos 26
MOTOR 61
MSX-DOS 9
Música 155

NEG 86
NEW 61
NOP 92
Notacion de complemento de dos 78
Notacion hexadecimal 79

OCT$ 61
ON ERROR GOTO 61
ON INTERVAL 61
ON INTERVAL GOSUB 36.61
ON KEY GOSUB 61
ON SPRITE GOSUB 35,62
ON STOP GOSUB 36,62
ON STRIG GOSUB 62
ON.. GOTO/GOSUB 61

OPEN 62
Operaciones de bifurcación y subrutinas del Z-80 88
Operaciones de bits 88
Operaciones de carga 84
Operaciones de E/S 96
Operaciones de intercambio de registros 92
Operaciones de pila 93
Operaciones de rotación y desplazamiento 87
Operaciones Lógicas 79
Operaciones Lógicas del Z-80 86
OR 80
Organización de la memoria 10
OUT 63
PAD 63
PAINT 63
PDL 63
PEEK 63
Pila 92
PLAY 64
POINT 64
POKE 64
POS 64
PRESET 64
PRINT 64
PRINT USING 65
Prioridad de sprite 28
Proceso de interrupciones 93
Programa Ejemplo: Diseño de sprites 37
Programa Ejemplo: Juego de caracteres 31
Programa Ejemplo: Tablero de apuntes 42
Programacion del VDP 124
Programas en ensamblador 83
PSET 65
PSG 22
Puertos de E/S 12
PUT SPRITE 65
RAM de video 13
Ranura 10
Ranura primaria 10
Raquetas 160
READ 66
Registros de la CPU 75
REM 66
RENUM 66
Representación binaria 77
RESTORE 66
RESUME 66
RET 90
RETI 90
RIGHT$ 67
RND 67
RUN 67

Salida de impresora 67
SAVE 67
SCREEN 68
SGN 68
SIN 68

Sistema Operativo de Disco 9
SOUND 68
SPACE$ 69
SPC 69
SPRITE ON 69
SPRITE ON/OFF/STOP 69
SPRITE STOP 69
SPRITE$ 69
Sprites 20,32
SQU 69
STICK 69
STOP 70
STOP ON/OFF/STOP 70
STR$ 70
STRIG 70
STRING$ 70
SWAP 71

TAB 66
Tabla de atributos de sprite 21
Tabla de color 18
Tabla de nombres de patrones 16
Tabla de patrones de sprite 21
Tabla de patrones generadores 16
Tablero digitalizador 161
TAN 71
Texto Multicolor 21
TIME 71
Transferencia y busqueda de bloques 90
TRON/TROFF 71
Unidad central de proceso 9
Unidad Logico-aritmetica 81
USR 67
Utilización de la RAM 107

VAL 72
Variables 24
Variables del sistema 25
VARPTR 72
VDP 13
Verificación de archivos de programa 51
Visualización de las teclas de función 58
VPEEK 73
VPOKE 73

WAIT 73
WIDTH 73

XOR 80